Scientific Application Performance on Leading Scalar and Vector Supercomputing Platforms

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

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

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: magnetic fusion (GTC), plasma physics (LBMHD3D), astrophysics (Cactus), and material science (PARATEC). We compare performance of the vector-based Cray X1, X1E, Earth Simulator, NEC SX-8, 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: 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; the highest per processor performance (by far) achieved by the full-production version of the Cactus ADM-BSSN; 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.

Keywords

Performance evaluation, parallel vector architecture, Lattice Boltzmann, particle-in-cell, Density Functional Theory, method of lines.

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 (Keyes et al. 2004). 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. A number of related investigations (Agarwal et al. 2004; Dunigan Jr. et al. 2003; Nakajima 2003; Pohl et al. 2004; Rabenseifner et al. 2005; Saini et al. 2006; Vetter et al. 2006) compare scalar and vector system performance in the context of microbenchmarks, numerical kernels, and full scale applications. Our study augments these efforts and examines the behavior of four diverse scientific applications with the potential to run at ultra-scale, in the areas of magnetic fusion (GTC), plasma physics (LBMHD3D), astrophysics (CACTUS) and material science (PARATEC). We compare the performance of leading commodity-based superscalar platforms utilizing the IBM Power3, Intel Itanium2, and AMD Opteron processors, with modern parallel vector systems: the Cray X1, Cray X1E, Earth Simulator (ES), and the NEC SX-8. 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 (Oliker et al. 2004; Oliker et al. 2005; Oliker et al. 2005) and makes several significant contributions: 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; the highest per processor performance (by far) achieved by the full-production version of the Cactus ADM-BSSN; 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

Name/ Location	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
Seaborg	Power3	SP Switch2	16	375	0.7	0.4	0.26	16.3	0.13	Fat-tree
Thunder	Itanium2	Quadrics	4	1400	5.6	1.1	0.19	3.0	0.25	Fat-tree
Jacquard	Opteron	InfiniBand	2	2200	4.4	2.3	0.51	6.0	0.59	Fat-tree
Phoenix	X1	Custom	4	800	12.8	14.9	1.16	7.1	6.3	4D-Hcube
Phoenix	X1E	Custom	4	1130	18.0	9.7	0.54	5.0	2.9	4D-Hcube
ESC	ES	Custom (IN)	8	1000	8.0	26.3	3.29	5.6	1.5	Crossbar
HLRS	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. The peak performance of the vector platforms does not include the scalar units or the divide/sqrt units of the ES/SX8.

In this section we briefly describe the computing platforms and scientific applications examined in our study. Table 2 presents an overview of the salient features for the six parallel HEC architectures, including:

- STREAM (J.D. McCalpin 2006) benchmark results (Stream BW), shows the measured (optimized) EP-STREAM (Luszczek et al. 2005) 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 for 8-byte message (Uehara et al. 2003; Dunigan Jr. 2006).
- Measured bidirectional MPI bandwidth per processor pair when each processor in one node simultaneously exchanges data with a distinct processor in another node*.

*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.

Table 2 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 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 (Dunigan Jr. et al. 2005) 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 — our work presents results in MSP mode, as this is the most common paradigm for X1 computation. The four SSPs share a 2-way set associative 2MB data Ecache, a 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 as it was upgraded to an X1E in July 2005.

We also examine performance of the recently-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. The 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. All reported X1 and X1E results are in MSP mode.

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 (Habata et al. 2004). 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 recently-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

Name	Lines	Discipline	Methods	Structure
GTC	5,000	Magnetic Fusion	Particle in Cell, gyrophase-averaged Vlasov-Poisson	Particle/Grid
LBMHD3D	1,500	Plasma Physics	Magneto-Hydrodynamics, Lattice Boltzmann	Lattice/Grid
CACTUS	84,000	Astrophysics	Einstein Theory of GR, ADM-BSSN, Method of Lines	Grid
PARATEC	50,000	Material Science	Density Functional Theory, Kohn Sham, FFT	Fourier/Grid

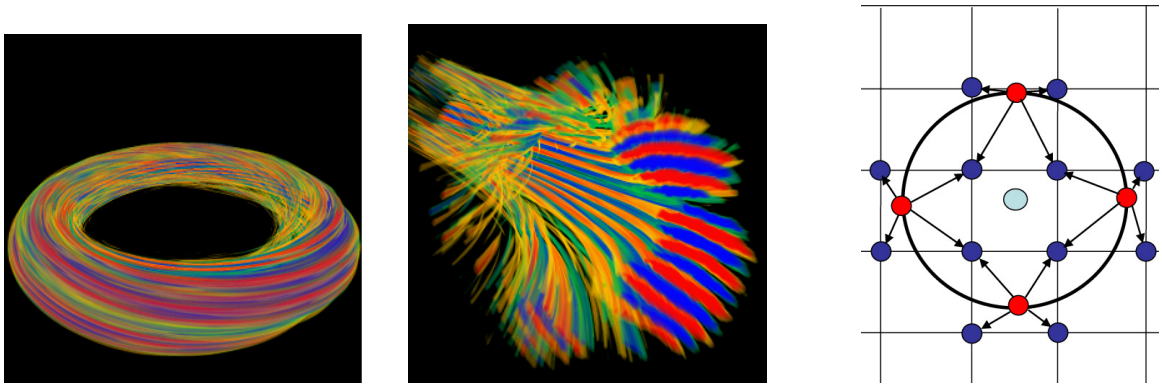
Table 2: Overview of scientific applications examined in our study.

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. We examine: 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; Cactus, an astrophysics code that evolves Einsteins equations from the Theory of General Relativity using the Arnowitt-Deser-Misner method; 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 2.1.

These codes represent candidate ultra-scale applications that have the potential to fully utilize leadership-class computing systems. Note that this set of applications have been designed and highly-optimized on superscalar platforms — thus, we only examine and describe newly devised optimizations for the vector platforms. For GTC, LBMHD3D, and Cactus we examine a weak-scaling configuration where the problem size grows with increasing concurrency; a fixed problem size (strong scaling) is investigated for PARATEC. 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 GTC: Turbulent Transport in Magnetic Fusion

GTC is a 3D particle-in-cell code used for studying turbulent transport in magnetic fusion plasmas (Lin et al. 1998; Ethier et al. 2005). The simulation geometry is that of a torus (see Figure 1), 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. Particles and waves interact self-consistently with each other, exchanging energy that grows or damps their motion and amplitude respectively. The particle-in-cell (PIC) method describes this complex phenomenon by solving the 5D gyrophase-averaged kinetic equation coupled to the Poisson equation.



(a) (b) (c)

Figure 1: Advanced volume visualization of the electrostatic potential field created by the plasma particles in a GTC simulation, showing (a) the whole volume and (b) a cross-section through a poloidal plane where the elongated eddies of the turbulence can be seen*. (c) The charge deposition approach in GTC's 4-point averaging scheme.

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 charge scattering 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. In the classic PIC method a point particle is followed directly and its charge is distributed to its nearest neighboring grid points. However, in the GTC gyrokinetic PIC approach, the fast circular motion of the charged particle around the magnetic field lines is averaged out and replaced by a charged ring. The 4-point average method (Lee 1987) consists of picking four points on that charged ring, each one having a fraction of the total charge, and distributing that charge to the nearest grid points (see Figure 1c). In this way, the full influence of the fast, circular trajectory is preserved without having to resolve it. However, this methodology inhibits vectorization since multiple particles may concurrently attempt to deposit their charge onto the same grid point.

The memory-dependency conflicts during the charge deposition phase can be avoided by using the work-vector method (Oliker et al. 2004), 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.

3.1 Particle Distribution 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 3, 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 (Oliker et al. 2004) 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

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

P	Part/ Cell	Seaborg Power3 Gflop/P	%Pk	Thunder Itanium2 Gflop/P	%Pk	Jacquard Opteron Gflop/P	%Pk	Phoenix X1 (MSP) Gflop/P	%Pk	Phoenix X1E (MSP) Gflop/P	%Pk	ESC ES Gflop/P	%Pk	HLRS SX-8 Gflop/P	%Pk
64	100	0.14	9	0.39	7	0.59	13	1.29	10	1.67	9	1.60	20	2.39	15
128	200	0.14	9	0.39	7	0.59	13	1.22	10	1.67	9	1.56	20	2.28	14
256	400	0.14	9	0.38	7	0.57	13	1.17	9	1.64	9	1.55	19	2.32	15
512	800	0.14	9	0.38	7	0.51	12	—	—	1.57	9	1.53	19	—	—
1024	1600	0.14	9	0.37	7	—	—	—	—	1.50	8	1.88	24	—	—
2048	3200	0.13	8	0.37	7	—	—	—	—	—	—	1.82	23	—	—
4096	6400	—	—	—	—	—	—	—	—	—	—	1.76	22	—	—

Table 3: GTC performance on each of the studied architectures using a fixed number of particles per processor.

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 total, 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 processors and, if needed, to considerably increase the number of particles in the simulation. The updated 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 (Lee 1987). 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 new parallel algorithm.

3.2 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 number of particles per processor, where each processor follows about 3.2 million particles. Table 3 shows the performance results for the six architectures in our study. The first striking difference from the previous GTC vector study (Oliker et al. 2004), is the considerable increase in concurrency. The new particle distribution algorithm allowed GTC to efficiently utilize 4,096 MPI processes (compared with only 64 using the previous approach), although this is not the limit of its scalability as recent Blue Gene/L benchmarks have shown excellent scaling past the 30,000-processor mark. With this new algorithm in place, GTC fulfilled the very strict scaling requirements of the ES and achieved an unprecedented 7.2 Tflop/s on 4,096 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.

Further vector optimizations were explored when GTC was ported to the Cray X1 at ORNL and later to its upgrade, the Cray X1E. The ES optimizations did not perform as well on the X1/X1E due to a much slower scalar processor and the large impact of having sections of the codes that neither multistreamed nor vectorized. Such sections run at least 32 times slower than fully multistreamed and vectorized sections on the X1/X1E. Therefore new optimizations

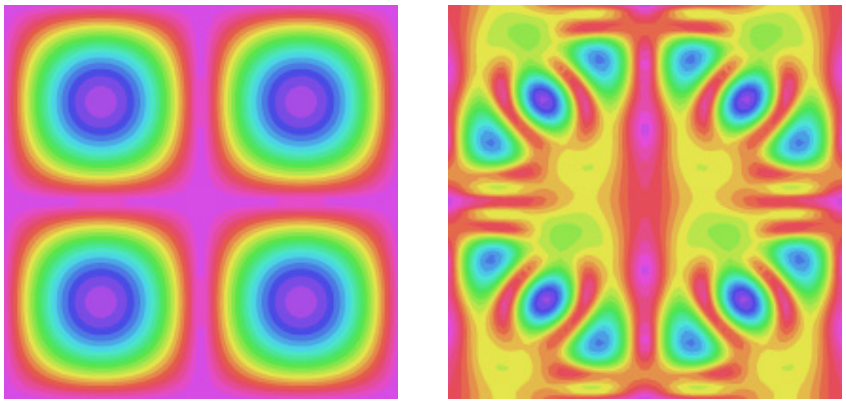
were employed on the X1E platform, including the vectorization of a few smaller loops in the code, and reordering the array dimensions for several GTC grid arrays. This latter change improved the performance of the code by 20% on the X1E but resulted in only a marginal 3% speedup on the ES. This discrepancy is most likely attributed to the faster memory system of the ES. Recall that the X1E architecture increases memory contention by doubling the number of MSPs per MCM, and increases processor speed (compared with the X1), without a commensurate increase in memory performance. Nonetheless, the new vector and multistream optimizations allow GTC to run equivalently (in absolute terms) on the X1E MSP and ES processors; however, the ES efficiency is significantly higher (23% vs. 9%). It is worth mentioning that some of these latest vector optimizations have not been tested on the X1 due to its upgrade to X1E.

Of the superscalar systems tested, the AMD Opteron with InfiniBand cluster (Jacquard) gives impressive results. GTC achieved slightly 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.

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

4 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 (Succi 2001). 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) (Dellar 2002; Macnab et al. 2001) 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 2 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).



(a) (b)

Figure 2: 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 (Wellein et al. 2006) was implemented, saving on the work required by the stream step. They noticed that the 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.

4.1 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 (Wellein et al. 2006), 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.

4.2 Experimental Results

P	Grid Size	Seaborg Power3		Thunder Itanium2		Jacquard Opteron		Phoenix X1 (MSP)		Phoenix X1E (MSP)		ESC ES		HLRS 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	0.26	5	0.70	16	5.19	41	6.19	34	5.50	69	7.89	49
64	256^3	0.15	10	0.35	6	0.68	15	5.24	41	5.73	32	5.25	66	8.10	51
256	512^3	0.14	9	0.32	6	0.60	14	5.26	41	5.65	31	5.45	68	9.52	60
512	512^3	0.14	9	0.35	6	0.59	13	—	—	5.47	30	5.21	65	—	—
1024	1024^3	—	—	—	—	—	—	—	—	—	—	5.44	68	—	—

Table 4: 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.

Table 4 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 1024-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 absolute performance is similar to the ES. 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.

For the X1E the same strategy is followed with one addition: two arrays are marked as being unsuitable for caching by inserting compiler directives in the code. The logic behind this approach is to save cache space for data that can effectively be reused[‡]. This increased performance by roughly 10%. Even with this optimization, percentage of peak lags behind all the other vector architectures.

[‡]Patrick Worley, personal communication

Turning to the superscalar architectures, the Opteron-based Jacquard cluster outperforms the Itanium2-based Thunder by almost a factor of 2X. One source of this disparity is that the Opteron achieves a STREAMS (J.D. McCalpin 2006) bandwidth of more than twice that of the Itanium2 (see Table 2). 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, Seaborg (Power3) does quite reasonably, obtaining a higher percent of peak than Thunder, but falling behind Jacquard.

5 CACTUS: General Relativity 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 (CACTUS 2006; Alcubierre et al. 2000) is designed to evolve Einstein's 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 (Font et al. 2002), this study focuses exclusively on the GR solver, which implements the Arnowitt-Deser-Misner (ADM) Baumgarte--Shapiro-Shibata-Nakamura (BSSN) (Alcubierre et al. 2000) method for stable evolutions of black holes. Figure 3 presents a visualization of one of the first simulations of the grazing collision of two black holes computed by the Cactus code. The merging black holes are enveloped by their "apparent horizon", which is colorized by its Gaussian curvature. The concentric surfaces that surround the black holes are equipotential surfaces of the gravitational flux of the outgoing gravity wave generated by the collision.

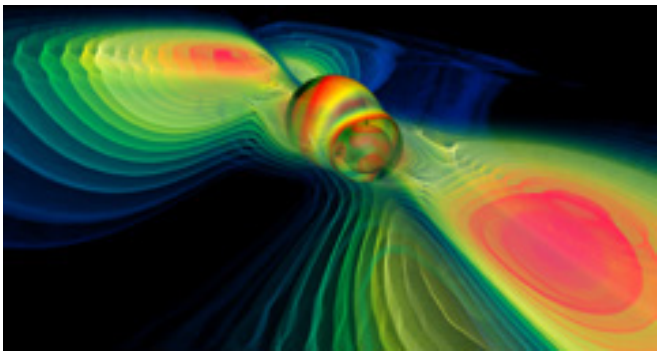


Figure 3: Visualization of grazing collision of two black holes as computed by Cactus*.

The Cactus General Relativity components solve Einstein's equations as an initial value problem that evolves partial differential equations on a regular grid using the method of finite differences. The core of the General Relativity solver uses the ADM formalism, also known also as the 3+1 form. For the purpose of solving Einstein's equations, the ADM solver decomposes the solution into 3D spatial hypersurfaces that represent different slices of space along the time dimension. In this formalism, 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 (Alcubierre et al. 2000). A "lapse" function describes the time slicing between hypersurfaces for each step in the evolution. A "shift metric" 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 grid is block domain decomposed so that each processor has a section of the global grid. The standard MPI driver for Cactus solves the PDE on a local grid section and then updates the values at the ghost zones by exchanging data on the faces of its topological neighbors in the domain decomposition.

*Visualization by Werner Benger (AEI/ZIB) using Amira {`http://www.amiravis.com`}.

5.1 Vectorization Details

For the superscalar systems, the computations on the 3D grid 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. On vector architectures these blocking optimizations were disabled, since they reduced the vector length and inhibited performance. The ES compiler misidentified some of the temporary variables in the most compute-intensive loop of the ADM-BSSN algorithm as having inter-loop dependencies. When attempts to force the loop to vectorize failed, a temporary array was created to break the phantom dependency.

Another performance bottleneck that arose on the vector systems was the cost of calculating radiation boundary conditions. The cost of boundary condition enforcement is inconsequential on the microprocessor based systems, however they unexpectedly accounted for up to 20% of the ES runtime and over 30% of the X1 overhead. The boundary conditions were vectorized using very lightweight modifications such as inline expansion of subroutine calls and replication of loops to hoist conditional statements outside of the loop. Although the boundaries were vectorized via these transformations, the effective vector length remained infinitesimally small. Obtaining longer vector lengths would have required more drastic modifications that were deemed impractical due the amount of the Cactus code that would be affected by the changes. This modification was very effective on the X1 because the loops could be multistreamed. Multistreaming enabled an easy 3x performance improvement in the boundary calculations that reduced their runtime contribution from the most expensive part of the calculation to just under 9% of the overall wallclock time. These same modifications produced no net benefit for the ES or SX-8, however, because the extremely short vector lengths.

5.2 Experimental Results

The full-fledged production version of the Cactus ADM-BSSN application was run on each of the architectures with results for two grid sizes shown in Table 5. 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.

P	Grid Size	Seaborg Power3		Thunder Itanium2		Jacquard Opteron		Phoenix X1 (MSP)		ESC ES		HLRS SX-8	
		Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk
16	80^3	0.31	21	0.60	11	0.98	22	0.54	4	1.47	18	1.86	12
64	80^3	0.22	14	0.58	10	0.81	18	0.43	3	1.36	17	1.81	11
256	80^3	0.22	14	0.58	10	0.76	17	0.41	3	1.35	17	1.75	11
1024	80^3	0.22	14	—	—	—	—	—	—	1.34	17	—	—
16	$250\text{x}64^2$	0.10	6	0.58	10	0.82	19	0.81	6	2.83	35	4.27	27
64	$250\text{x}64^2$	0.08	6	0.58	10	0.92	21	0.72	6	2.70	34	4.04	25
256	$250\text{x}64^2$	0.07	5	0.57	10	0.68	16	0.68	5	2.70	34	3.87	24
1024	$250\text{x}64^2$	0.06	4	—	—	—	—	—	—	2.70	34	—	—

Table 5: Cactus weak-scaling performance using per-processor grid sizes of 80x80x80 and 250x64x64.

For the vector systems, Cactus achieves almost perfect vector operation ratio (over 99%) while the vector length 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 (vector length = 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. Also, note that for the Cactus simulations, bank conflict overheads are negligible for the chosen (not power of two) grid sizes.

Recall 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 different dimension of architectural balance that is specific to 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

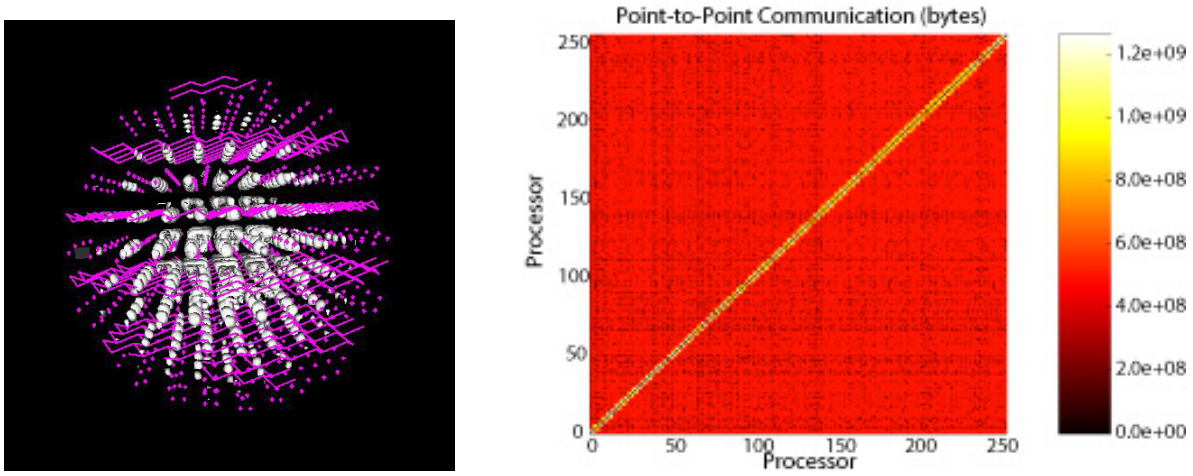
was particularly acute on the X1, which suffered a much greater impact from unvectorized code than the ES. On the SX-8, the boundary conditions occupy approximately the same percentage of the execution time as it did on the ES, which is consistent with the fact that the performance improvements in the SX-8 scalar execution unit have scaled proportionally with the vector performance improvements. The decreased execution efficiency is primarily reflected in lower efficiency in the vector execution. (X1E results are not available, as the code generated by the new compilers on the X1E exhibit runtime errors that are still being investigated by Cray engineers.)

The microprocessor based systems offered lower peak performance and generally lower efficiency than the NEC vector systems. Jacquard, however, offered impressive efficiency as well as peak performance in comparison to Seaborg and Thunder. Unlike Seaborg, Jacquard maintains its performance even for the larger problem size. The relatively low scalar performance on the microprocessor-based systems is partially due to register spilling, which is caused by the large number of variables in the main loop of the BSSN calculation. However, the much lower memory latency of the Opteron and higher effective memory bandwidth relative to its peak performance allow it to maintain higher efficiency than most of the other processors.

6 PARATEC: First Principles Materials Science

PARATEC (PARAllel Total Energy Code (PARATEC 2006)) 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.



(a) (b)

Figure 4: (a) Conduction band minimum electron state for a CdSe quantum dot of the type used in PARATEC experiments (Wang 2001). (b) Volume of point to point communication of PARATEC using 256 processors.

6.1 Communication Structure

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. 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. Much of the compute intensive part of the code is carried out in Fourier space so it is very important to load balance that part of the calculation. The sphere geometry is load balanced 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.

The bulk of PARATEC's communication occurs within 3D FFTs, involving three sets of one dimensional FFTs in the x,y,z direction and three sets of data transposes after each set of FFTs. The number of data transposes can be reduced to two if the data order at the end of the FFT is not required in the same decomposition as the initial data i.e. it is returned ordered z,y,x as is the case in the specialized 3D FFTs used in PARATEC. The two parallel data transposes then represent the data communication intensive parts of the code. The first transpose is non-local and involves communication of messages of similar sizes between all the processors. The second transpose is performed more locally to the processors by having, as closely as possible, complete planes of data on each processor. During this transpose processors will only communicate with neighboring processors. A more detailed description of the 3D FFTs can be found in (Canning et al. 2000). Figure 4(b) shows the data communication pattern for PARATEC obtained using IPM (Skinner 2005) profiling. The even background color represents the first all-to-all transpose and the color on the diagonal corresponds to the local communications from the second transpose.

6.2 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 cut-off 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 (Oliker et al. 2004) examined smaller physical systems at lower concurrencies. Figure 4(a) shows an example of the computed conduction band minimum electron state for a CdSe quantum dot.

P	Seaborg Power3		Thunder Itanium2		Jacquard Opteron		Phoenix X1 (MSP)		Phoenix X1E (MSP)		ESC ES		HLRS 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	63	—	—	—	—	4.25	33	4.88	27	—	—	7.91	49
128	0.93	62	2.84	51	—	—	3.19	25	3.80	21	5.12	64	7.53	47
256	0.85	57	2.63	47	1.98	45	3.05	24	3.24	18	4.97	62	6.81	43
512	0.73	49	2.44	44	0.95	22	—	—	2.22	12	4.36	55	—	—
1024	0.60	40	1.77	32	—	—	—	—	—	—	3.64	46	—	—
2048	—	—	—	—	—	—	—	—	—	—	2.67	33	—	—

Table 6: PARATEC results using 488 atom CdSe quantum dot on the evaluated platforms. X1E experiments were conducted using X1-compiled binary.

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.

Results on the X1E were obtained by running the binary compiled on the X1, as running with an optimized X1E generated binary (-O3) caused the code to freeze (Cray engineers are investigating the problem). The results on the X1E are on average 14% faster than the X1, resulting in slightly lower percentage of peak. This decrease in efficiency is probably due to the additional memory and interconnect contention on the X1E platform (see Table 2).

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 Seaborg is primarily due to the increased communication cost at high concurrencies. The Thunder and Jacquard systems show a similar high percentage of peak at lower concurrencies, giving Thunder a higher absolute performance than the Jacquard 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 linear algebra computations due to its lack of floating-point multiply add (FMA) hardware. The Quadrics-based Thunder 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. Overall all the architectures studied here would obtain better scaling at higher concurrency by running a larger physical system.

7 Summary and Conclusions

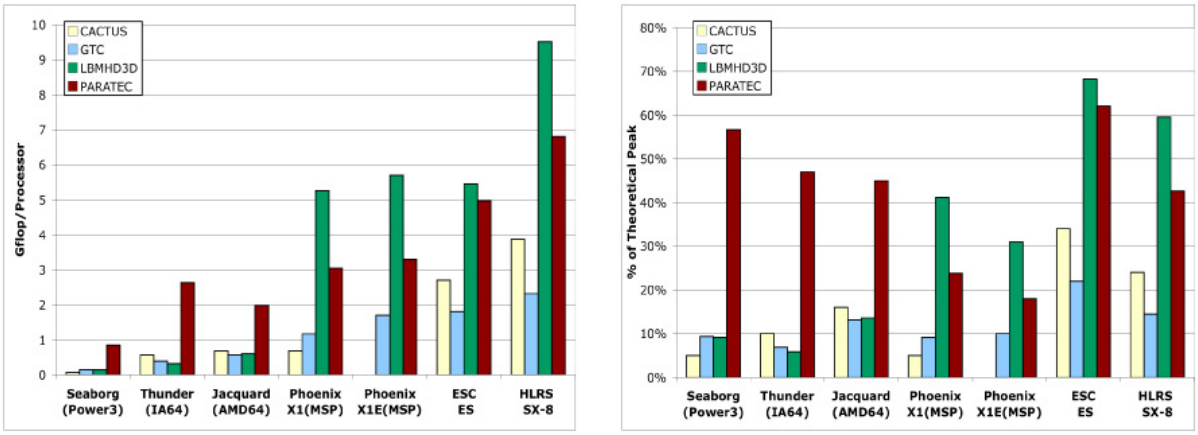


Figure 5: Overview of performance for the four studied applications on 256 processors, comparing (left) absolute speed and (right) percentage of theoretical peak.

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 important contributions. We 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 introduce, 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. We also demonstrate the highest sustained and aggregate performance ever achieved on the Cactus astrophysics simulation of the General Relativity calculation. 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 5 presents a performance overview for the four studied applications using 256 processors, comparing (left) absolute speed[†] and (right) percentage of theoretical peak. Overall results show that the SX-8 impressively achieves the best per-processor performance — outperforming all other architectures in our study (and all architectures examined to date). In terms of sustained efficiency, however, the ES consistently attains the highest percentage across all evaluated platforms. The SX-8 cannot 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 systems achieve a higher fraction of peak compared than the X1/X1E for our examined codes, frequently by a significant margin. This is due, in part, to superior scalar processor performance and superior memory behavior. However, in terms of absolute runtime, the X1(E) outperforms the evaluated superscalar systems for most of the applications examined in our study. Results also show that, in many cases, the X1E percentage of peak is somewhat lower than the X1, due 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.

Finally, a comparison of the modern superscalar platforms shows that Jacquard dramatically outperforms the Thunder system for GTC, LBMHD3D, and Cactus with the situation reversed for PARATEC. 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-based Jacquard system also benefits from its 2-way SMP node that has STREAM bandwidth performance more than twice that of the Itanium2-based Thunder, which utilizes a 4-way SMP node configuration.

PARATEC, on the other hand, relies on global data transpositions, giving the Quadrics-based Thunder system a performance advantage over the InfiniBand-based Jacquard 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 Seaborg's (relatively old) IBM Power3 architecture consistently achieves a higher fraction of peak than the Thunder system and shows good scaling performance across our entire application suite.

Our overall analysis points to several architectural features that could benefit future system designers. First, as seen by the relatively modest gains between the X1 and X1E processors, vector system architects must continue to increase both scalar processor and memory performance proportionally with improvements to the vector units. Otherwise, further improvements to the peak vector performance will likely deliver diminishing returns when averaged across the requirements of the broader scientific workload. Results also show that superscalar designers must continue to focus on effectively hiding main memory latency for applications with low computational intensity (such as CACTUS, GTC, and LBMHD). Recent design trends of moving memory controllers on-broad the processor chip — as seen on the Opteron — help address this deficiency, but significant challenges still remain. For example, the irregular access patterns exhibited by the charge deposition phase of GTC make the cache hierarchy less effective for reducing the average latency of memory accesses, however the potential for conflicts in the update phase in GTC presents a non-optimal solution for the vector approach. Future architecture could mitigate some of these effects by supporting either atomic memory updates or explicit software-controlled local store (scratchpad) memory. A transactional memory approach, is one promising methodology that could potentially improve the efficiency of the charge deposition phase

[†]Recall that the ratio between the computational rates is the same as the ratio of runtimes across the evaluated systems.

of PIC codes without requiring extensive code re-implementation. Finally, in terms of interconnect design, PARATEC results show that support for low-latency high-bisection bandwidth networks is essential for spectral and plane-wave DFT codes that rely on global, multidimensional FFTs.

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 and well as the early SX-8 system access provided by HLRS, Germany. The authors also sincerely thank Patrick Worley and David Parks for their many useful suggestions for vector optimizations strategies, 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. 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-AC02-05CH11231. 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-AC02-05CH11231. Dr. Ethier was supported by the Department of Energy under contract number DE-AC020-76-CH-03073.

References

Agarwal, P. A. et al. (May 17-21, 2004). Cray X1 evaluation status report. In *Proc. of the 46th Cray Users Group Conference*.

Alcubierre, M., G. Allen, B. Brügmann, E. Seidel, and W.-M. Suen (2000). Towards an understanding of the stability properties of the 3+1 evolution equations in general relativity. *Phys. Rev. D 62*, 124011.

CACTUS (2006). Cactus Code Server. `http://www.cactuscode.org`.

Canning, A., L. Wang, A. Williamson, and A. Zunger (2000). Parallel empirical pseudopotential electronic structure calculations for million atom systems. *J. Comp. Phys. 160*(1), 29–41.

Dellar, P. J. (2002). Lattice kinetic schemes for magnetohydrodynamics. *J. Comput. Phys. 79*(1), 95–126.

Dunigan Jr., T. (2006). ORNL Cray X1 Evaluation. `http://www.csm.ornl.gov/~dunigan/cray`.

Dunigan Jr., T. H., M. R. Fahey, J. B. W. III, and P. H. Worley (2003). Early evaluation of the Cray X1. In *Proceedings of the ACM/IEEE SC2003 Conference on High Performance Networking and Computing, 15-21 November 2003, Phoenix, AZ, USA, CD-Rom*. ACM.

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

Ethier, S., W. Tang, and Z. Lin (2005). Gyrokinetic particle-in-cell simulations of plasma microturbulence on advanced computing platforms. *J. Phys. : Conf. Series 16*(1), 1–15.

Font, J. A., M. Miller, W. M. Suen, and M. Tobias (2002, Apr). Three dimensional numerical general relativistic hydrodynamics: Formulations, methods, and code tests. *Phys. Rev. D 65*(8), 084024.

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

J.D. McCalpin (2006). J.D. McCalpin. STREAM: Measuring sustainable memory bandwidth in high performance computers, 1995. `http://www.cs.virginia.edu/stream`.

Keyes, D. et al. (2004). A Science-Based Case for Large-Scale Simulation (SCALES). `http://www.pnl.gov/scales`.

Lee, W. W. (1987). Gyrokinetic particle simulation model. *J. Comp. Phys. 72*(1), 243–269.

Lin, Z., T. S. Hahm, W. W. Lee, W. M. Tang, and R. B. White (1998). Turbulent transport reduction by zonal flows: Massively parallel simulations. *Science 281*(5384), 1835–1837.

Luszczek, P., J. Dongarra, D. Koester, R. Rabenseifner, J. K. R. B. Lucas, et al. (2005). Introduction to the HPC Challenge Benchmark Suite. `http://icl.cs.utk.edu/hpcc/index.html`.

Macnab, A., G. Vahala, P. Pavlo, , L. Vahala, and M. Soe (2001). Lattice boltzmann model for dissipative incompressible MHD. In *Proc. 28th EPS Conference on Controlled Fusion and Plasma Physics*, Volume 25A.

Nakajima, K. (March 27-30, 2003). 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.

Oliker, L., A. Canning, J. Carter, J. Shalf, and S. Ethier (2004). Scientific computations on modern parallel vector systems. In *Proceedings of the ACM/IEEE SC2004 Conference on High Performance Networking and Computing, 6-12 November 2004, Pittsburgh, PA, USA, CD-Rom*. IEEE Computer Society.

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

Oliker, L., J. Carter, M. Wehner, A. Canning, S. Ethier, B. Govindasamy, A. Mirin, and D. Parks (2005). Leading computational methods on scalar and vector hec platforms. In *Proceedings of the ACM/IEEE SC2005 Conference on High Performance Networking and Computing, November 12-18, 2005, Seattle, WA, USA, CD-Rom*. IEEE Computer Society.

PARATEC (2006). PARAllel Total Energy Code. `http://www.nersc.gov/projects/paratec`.

Pohl, T., F. Deserno, N. Thurey, U. Rude, P. Lammers, G. Wellein, and T. Zeiser (2004). Performance evaluation of parallel large-scale lattice bolzmann applications on three supercomputing architectures. In *Proceedings of the ACM/IEEE SC2004 Conference on High Performance Networking and Computing, 6-12 November 2004, Pittsburgh, PA, USA, CD-Rom*. IEEE Computer Society.

Rabenseifner, R., S. R. Tiyyagura, and M. Müller (Sep 18-21, 2005). Network Bandwidth Measurements and Ratio Analysis with the HPC Challenge Benchmark Suite (HPCC). In *EURO PVM MPI: 12th European Parallel Virtual Machine and Message Passing Interface Conference*, Sorrento, Italy.

Saini, S., R. Ciotti, T. N. Gunney, T. E. Spelce, A. Koniges, et al. (April 25-29, 2006). Performance evaluation of supercomputers using hpcc and imb benchmarks. In *Proc. 5th International Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS)*, Rhodes Island, Greece.

Skinner, D. (2005). Integrated Performance Monitoring: A portable profiling infrastructure for parallel applications. In *Proc. ISC2005: International Supercomputing Conference*, Heidelberg, Germany.

Succi, S. (2001). *The Lattice Boltzmann Equation For Fluids and Beyond*. Oxford University Press.

Uehara, H., M. Tamura, and M. Yokokawa (2003). MPI performance measurement on the Earth Simulator. Technical Report # 15, NEC Research and Development.

Vetter, J., S. Alam, T. Dunigan, Jr., M. Fahey, P. Roth, and P. Worley (April 25-29, 2006). Early evaluation of the Cray XT3. In *Proc. IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, Rhodes Island, Greece.

Wang, L. (2001). Calculating the influence of external charges on the photoluminescence of a CdSe quantum dot. *J. Phys. Chem. 105*, 2360.

Wellein, G., T. Zeiser, S. Donath, and G. Hager (2006). On the single processor performance of simple lattice bolzmann kernels. *Computers and Fluids 35*, 910.