Performance Evaluation of Scientific Applications on Modern Parallel Vector Systems

Jonathan Carter, Leonid Oliker, John Shalf

NERSC/CRD, Lawrence Berkeley National Laboratory, Berkeley, CA 94720
{jtcarter,loliker,jshalf}@lbl.gov

Abstract. Despite their dominance of high-end computing (HEC) through the 1980's, vector systems have been gradually replaced by microprocessor-based systems. However, while peak performance of microprocessor-based systems has grown exponentially, the gradual slide in sustained performance delivered to scientific applications has become a growing concern among HEC users. Recently, the Earth Simulator and Cray X1/X1E parallel vector processor systems have spawned renewed interest in vector technology for scientific applications. In this work, we compare the performance of two lattice-Boltzmann applications and the Cactus astrophysics package on vector based systems including the Cray X1/X1E, Earth Simulator, and NEC SX-8, with commodity-based processor clusters including the IBM SP Power3, IBM Power5, Intel Itanium2, and AMD Opteron processors. We examine these important scientific applications to quantify effective performance and investigate if the efficiency benefits are applicable to a broader array of numerical methods.

1 Introduction

Despite their dominance of high-end computing (HEC) through the 1980's, vector systems have been progressively replaced by microprocessor based systems due to the lower costs afforded by mass-market commercialization and the relentless pace of clock frequency improvements for microprocessor cores. However, while peak performance of superscalar systems has grown exponentially, the gradual slide in sustained performance delivered to scientific applications has become a growing concern among HEC users. 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. Furthermore, now that power dissipation is limiting the growth rate in clock frequency, the low sustained performance of superscalar systems has risen to the forefront of concerns. The latest generation of custom-built parallel vector systems have the potential to address these performance challenges for numerical algorithms amenable to vectorization.

The architectural complexity of superscalar cores has grown dramatically in the past decade in order to support out-of-order execution of instructions

that feed an increasing number of concurrent functional units. However, there is growing evidence that despite the enormously complex control structures, typical superscalar implementations are only able to exploit a modest amount of ILP—on the order of 1-1.5 operations per cycle on typical scientific applications. Vector technology, by contrast, is well suited to problems with plenty of inherent data parallelism. For such problems, the vector approach reduces control complexity because each operation defined in the instruction stream implicitly controls numerous functional units operating in tandem, allowing memory latencies to be masked by overlapping pipelined vector operations with memory fetches.

However, when such operational parallelism cannot be found, the efficiency of the vector architecture can suffer from the properties of Amdahl's Law, where the time taken by the portions of the code that are non-vectorizable can easily dominate the execution time. In this regard, modern vector machines are quite unlike the Cray 1 [1] in that the scalar performance is well below average compared to commodity systems targeted at business applications. It is difficult for vector vendors to compete on scalar processor performance, as the enormous technology investment necessary to keep pace with the microprocessors is too great to sustain without a large market share. Thus today's vector systems have been unable to produce competitive scalar processor implementations, resulting in more significant performance penalties for non-vectorizable code portions when compared to classic vector system implementations.

In the context of these evolving architectural changes, it is important to continue the assesment of vector platforms in the face of increasing algorithm complexity. For this reason, our study focuses on full applications to get more realistic assessments of vector performance in the face of limitations imposed by increased application complexity. This work compares performance between the vector-based Cray X1/X1E, Earth Simulator (ES) and NEC SX-8, with commodity-based superscalar platforms: Intel Itanium2, AMD Opteron, and the IBM Power3 and Power5 systems. We study the behavior of three scientific codes with the potential to run at ultra-scale: lattice-Boltzmann (LB) simulations of fluid dynamics and magnetohydrodynamics (LBMHD3D and ELBM3D), and astrophysics (CACTUS). Our work builds on our previous efforts [2, 3] and makes the contribution of adding recently acquired performance data for the SX-8, and the latest generation of superscalar processors. Additionally, we explore improved vectorization techniques Cactus boundary conditions, and effect of cache-bypass pragmas for the LB applications. Overall results show that the SX-8 attains unprecedented aggregate performance across our evaluated applications, continuing the trend set by the ES in our previous performance studies. Our study also shows that the slide in sustained performance of microprocessor cores is not irreversible if microprocessor architectures are willing to invest the effort to make architectural decisions that eliminate bottlenecks for scientific applications. For instance, the Power5 shows some improvement over its predecessors (the Power3 and Power4) through more advanced latency-hiding prefetch features and a much higher bandwidth to main memory.

2 HEC Platforms and Evaluated Applications

In this section we briefly describe the computing platforms and scientific applications examined in our study. Tables 1 and 2 present an overview of the salient features for the eight parallel HEC architectures. Observe that the vector machines have higher peak performance and better system balance than the superscalar platforms. Additionally, the X1, ES, SX-8, and to a lessor extent the X1E, have high memory bandwidth (as measured by HPCC EP Stream [4]) relative to peak CPU speed (bytes/flop), allowing them to more effectively feed the arithmetic units. Finally, the vector platforms utilize interconnects that are tightly integrated to the processing units, with high performance network buses and low communication software overhead.

Four superscalar commodity-based platforms are examined in our study. The IBM Power3 experiments reported were conducted on the 380-node pSeries system, running AIX 5.2 (xlf compiler 8.1.1) and located at Lawrence Berkeley National Laboratory (LBNL). Each SMP node consists of sixteen 375 MHz processors (1.5 Gflop/s peak) connected to main memory via the Colony switch using an omega-type topology. The Power5 system, also located at LBNL's NERSC facility, consists of 111 8-way Power5 nodes operating at 1.9 GHz (7.6 GFlops/s) and interconnected by a dual-plane Federation interconnect using a fat-tree/CLOS topology. Like the Power3, this system also runs AIX 5.2, but uses the newer xlf 9.1 compiler. The AMD Opteron system, is also located at LBNL and contains 320 dual nodes, running 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 Itanium experiments were performed on the 1024 node system located at Lawrence Livermore National Laboratory. Each node contains four 1.4 GHz Itanium2 processors (5.6 Gflop/s peak) and runs 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/X1E is designed to combine traditional vector strengths with the generality

Table 1. CPU overview of the Power3, Power5, Itanium2, Opteron, X1/X1E, ES, and SX-8 platforms.

Platform	CPU/ Node	Clock (MHz)	Peak (GF/s)	Stream BW (GB/s)	Peak (Byte/Flop)
Power3	16	375	1.5	0.4	0.3
Power5	8	1900	7.6	5.4	0.7
Itanium2	4	1400	5.6	1.1	0.2
Opteron	2	2200	4.4	2.3	0.5
X1	4	800	12.8	14.9	1.2
X1E	4	1130	18.1	9.7	0.5
ES	8	1000	8.0	26.3	3.3
SX-8	8	2000	16.0	41.0	2.6

and scalability features of modern superscalar cache-based parallel systems. The computational core, called the single-streaming processor (SSP), contains two 32-stage vector pipes running at 1130 MHz. Each SSP operates at 4.5 Gflop/s peak for 64-bit data. The SSP also contains a two-way out-of-order superscalar processor running at 400 MHz. Four SSP can be combined into a logical computational unit called the multi-streaming processor (MSP) with a peak of 18.0 Gflop/s. The four SSPs share a 2-way set associative 2 MB data Ecache, a unique feature for vector architectures that allows extremely high bandwidth (25–51 GB/s) for computations with temporal data locality. The X1E node consists of eight MSPs sharing a flat memory, and large system configuration are networked through a modified 2D torus interconnect. X1E nodes are partitioned into two logical 4-way SMP nodes from the application developers viewpoint. All reported X1E experiments were performed on the 1024-MSP system (several reserved for system services) running UNICOS/mp 3.1 (5.5 programming environment) and operated by Oak Ridge National Laboratory. The X1 experiments were performed on the 512-MSP system at ORNL prior to the upgrade to X1E.

The 1000 MHz Earth Simulator processor was the precursor to the NEC SX6, containing an 4-way replicated vector pipe with a peak performance of 8.0 Gflop/s per CPU. The system contains 640 ES nodes, 5120-processor, connected through a custom single-stage IN crossbar. The 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 2004 and 2005.

Finally, we examine the NEC SX-8, the world's most powerful vector processor. 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/SX6 predecessor, including improved divide performance, hardware square root functionality, and in-memory caching for reducing bank conflict overheads. However, the SX-8 used in our study uses commodity DDR-SDRAM; thus, we expect higher memory overhead

Table 2. Interconnect performance of the Power3, Power5, Itanium2, Opteron, X1, ES, and SX-8 platforms.

Platform	Network	MPI Lat (μsec)	MPI BW (GB/s/CPU)	Bisect BW (Byte/Flop)	Network Topology
Power3	Colony	16.3	0.13	0.09	Fat-tree
Power5	Federation	3.0	4.0	0.52	Fat-tree
Itanium2	Quadrics	3.0	0.25	0.04	Fat-tree
Opteron	InfiniBand	6.0	0.59	0.11	Fat-tree
X1	Custom	7.3	6.3	0.09	2D-torus
ES	Custom (IN)	5.6	1.5	0.19	Crossbar
SX-8	IXS	5.0	2.0	0.13	Crossbar

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 the half the peak of their vector counterparts. All reported SX-8 results were run on the 72 node 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

Two application domains from scientific computing were chosen to compare the performance of the vector-based and superscalar-based systems.

We examine LBMHD3D, a three-dimensional plasma physics application that uses the Lattice-Boltzmann method to study magneto-hydrodynamics [5]; ELBM3D, a a three-dimensional fluid dynamic application that uses the Lattice-Boltzmann method to study turbulent fluid flow [6]; and CACTUS, a modular framework supporting a wide variety of multi-physics applications [7], using the Arnowitt-Deser-Misner (ADM) formulation for the evolution of the Einstein equations from first principles that are augmented by the Baumgarte-Shapiro-Shibata-Nakamura (BSSN) [8] method to improve numerical stability for simulation of black holes.

These codes represent grand-challenge scientific problems that require access to ultrascale systems and provide code bases mature enough that they have the potential to fully utilize the largest-scale computational resources available. Performance results, presented in Gflop/s per processor and percentage of peak, are used to compare the relative time to solution of our evaluated computing systems. When different algorithmic approaches are used for the vector and scalar implementations, this value is computed by dividing a valid baseline flop-count by the measured wall-clock time of each platform.

3 Lattice-Boltzmann Turbulence Simulations

Lattice-Boltzmann methods (LBM) have proved a good alternative to conventional numerical approaches for simulating fluid flows and modeling physics in fluids [9]. The basic idea is to develop a simplified kinetic model that incorporates the essential physics, and reproduces correct macroscopic averaged properties. These algorithms have been used extensively over the past ten years for simulating Navier-Stokes flows, and more recently, several groups have applied the LBM to the problem of magneto-hydrodynamics (MHD) [10–12] with promising results [5]. As can be expected from explicit algorithms, LBM are prone to numerical nonlinear instabilities as one pushes to higher Reynolds numbers. These numerical instabilities arise because there are no constraints imposed to enforce the distribution functions to remain non-negative. Such entropic LBM algorithms, which do preserve the non-negativity of the distribution functions—even in the limit of arbitrary small transport coefficients—have recently been

developed for Navier-Stokes turbulence [13, 14]. Our LBM applications are representative of these two active research areas: the LBMHD3D code simulates the behavior of a conducting fluid evolving from simple initial conditions through the onset of turbulence; and the ELBM3D code uses the entropic LB algorithm to simulate the behaviour of Navier-Stokes turbulence.

While LBM methods lend themselves to easy implementation of difficult boundary geometries, e.g., by the use of bounce-back to simulate no slip wall conditions, here we report on 3D simulations under periodic boundary conditions, with the spatial grid and phase space velocity lattice overlaying each other. Each lattice point is associated with a set of mesoscopic variables, whose values are stored in vectors proportional to the number of streaming directions. The lattice is partitioned onto a 3-dimensional Cartesian processor grid, and MPI is used for communication. As in most simulations of this nature, ghost cells are used to hold copies of the planes of data from neighboring processors.

In simple terms an LB 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. However, a key optimization described by Wellein and co-workers [15] is often 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.

For ELBM3D, a non-linear equation must be solved for each grid-point and at each time-step so that the collision process satisfies certain constraints. The equation is solved via Newton-Raphson iteration (5 iterations are usually enough to converge to within 10^{-8}), and as this equation involves taking the logarithm of each component of the distribution function at each iteration, the whole algorithm become heavily constrained by the performance of the log function.

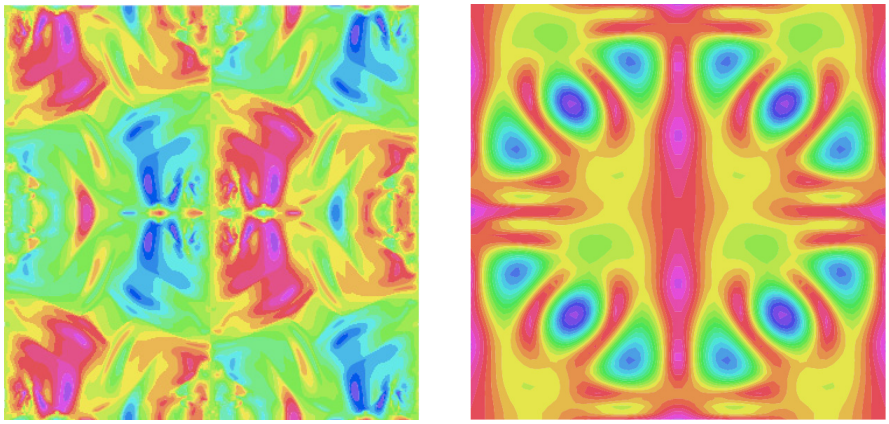
Figure 1 shows a slice through the xy-plane in the (left) ELBM3D and (right) LBMHD3D simulation, where the vorticity profile has distorted considerably after several hundred time steps as turbulence sets in.

3.1 Vectorization details

The basic computational structure consists of three nested loops over spatial grid points (typically 1000s iterations) with inner loops over velocity streaming vectors and, in the case of LBMHD3D, magnetic field streaming vectors (typically 10s iterations). Within these innermost loops the various macroscopic quantities and their updated values are calculated via various algebraic expressions.

For the LBMHD3D case, on both the ES and SX-8, the innermost loops were unrolled via compiler directives and the (now) innermost grid point loop was vectorized. This proved a very effective strategy, and was also followed on

Fig. 1. Contour plot of xy-plane showing the evolution of vorticity from well-defined tube-like structures into turbulent structures using (left) ELBM3D and (right) LBMHD3D.



the X1E. In the case of the X1E, however, the compiler needed more coercing via directives to multi-stream the outer grid point loop and vectorize the inner grid point loop once the streaming loops had been unrolled. Following Worley [16], we used the compiler directive NO_CACHE_ALLOC to attempt to optimize cache use on the X1E. This directive works by indicating to the compiler that certain arrays that have low reuse are not be to be cached. In this way, space is preserved for data that can be more beneficially cached. This optimization produced a speedup of 10%. For the superscalar architectures, we utilized a data layout that has been previously shown to be optimal on cache-based machines [15], but did not explicitly tune for the cache size on any machine.

For ELBM3D, in the case the vector architectures, the compilers were able to vectorize all loops containing log functions. The routine containing the non-linear equation solver was rewritten to operate on an array of grid points, rather than a single point, allowing vectorization of this recursive operation. After this optimization, high performance was achieved on all the vector systems. For the X1E, two other factors are important to note. In a similar way to LBMHD3D, compiler directives to enable efficient cache use led to a modest 5% speedup. Less data is moved per gridpoint in ELBM3D as compared with LBMHD3D, so cache tuning could reasonably be expected to produce less of a speedup. The call to the non-linear equation solving routine prevented multistreaming of the outer grid point loop on the X1E. Because of this, the innermost grid point loop is now both multistreamed and vectorized. For the tests run here, the vector length does not drop below 64, but it does lead to shorter vector lengths compared to the LBMHD3D code.

For the superscalar systems, using the rewritten non-linear equation solving routine proved to be much faster than the original approach. Presumably this

is due to a reduction of routine call overhead and better use of the functional units. Depending on the architecture, a speedup of 20-30% is achieved on switching to the new routine. Another important optimization was to use optimized library routines to compute a vector of logarithm values per invocation. Each architecture offers an optimized math function library: MASS for IBM Power5 and Power3, MKL for Intel Itanium2; and ACML for AMD Opteron. A 15-30% speedup over the the 'non-vector' log function is achieved, with the Itanium2 showing the largest speedup. In addition, the innermost grid point loop was blocked to try and improve cache reuse. A downside to this optimization is that it reduces the length of the array being passed to the log function. This produced very minor speedups for Power3 and Power5, a slowdown for the Itanium2, but a moderate improvement (roughly 15%)for the Opteron system.

3.2 Experimental Results

Tables 3 and 4 and present the performance of both LB applications across the seven architectures evaluated in our study. Cases where the memory or number of processors required exceeded that available are indicated with a dash.

Table 3. LBMHD3D performance in GFlop/s (per processor) across the studied architectures for a range of concurrencies and grid sizes. Percentage of peak is shown in parenthesis.

P	Size	Power3	Power5	Itanium2	Opteron	X1E	ES	SX-8
16	256^3	0.14 (9)	0.81 (11)	0.26 (5)	0.70 (16)	6.19 (34)	5.50 (69)	7.89 (49)
64	256^3	0.15 (10)	0.82 (11)	0.35 (6)	0.68 (15)	5.73 (32)	5.25 (66)	8.10 (51)
256	512^3	0.14 (9)	0.79 (10)	0.32 (6)	0.60 (14)	5.65 (31)	5.45 (68)	9.66 (60)
512	512^3	0.14 (9)	0.79 (10)	0.35 (6)	0.59 (13)	5.47 (30)	5.21 (65)	—

For LBMHD3D, observe that the vector architectures clearly outperform the scalar systems by a significant factor. Across these architectures, the application exhibits an average vector length (AVL) very close to the maximum (more than 255.5) and a vector operation ratio (VOR) of more than 99.5%. 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 — 65% even at the highest 512-processor concurrency. Examining the X1E behavior, we see that in MSP mode, absolute performance is between that of the ES and the SX-8. Although the SX-8 achieves the highest absolute performance, the percentage of peak is somewhat lower than that of ES. Based on previous work [17], we believe that this is related to the memory subsystem and use of DDR-SDRAM. 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 system achieves stream memory bandwidth (see Table 1) of more than twice that of the

Itanium2 system. 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. The Power5 achieves a slightly better percentage of peak than the Power3, but somewhat dissapointingly trails the Opteron.

For ELBM3D, all superscalar architectures achieve a high percentage of peak performance. The main reason is the much higher computational intensity and less complex data access patterns of the application relative to LBMHD3D. For the vector architectures, the converse is true—all achieve a lower percentage of peak, as compared to LBMHD3D, with the ES decreasing the most. The problem is not unvectorized code, the ELBM3D application has an AVL and VOR very close to that of LBMHD3D. Lack of arithmetic operations and data movement in the application has lessened the advantage of the fast ES memory and the log function is probably a bottleneck in computation. However, although the advantage of vector over superscalar has diminished, the SX-8 still achieves tha highest overall performance, followed by the X1E and ES.

Table 4. ELBM3D performance in GFlop/s (per processor) across the studied architectures for a range of concurrencies and grid sizes. Percentage of peak is shown in parenthesis.

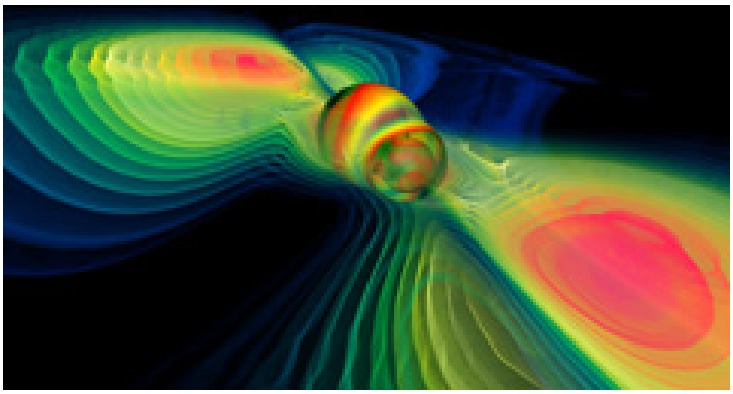
P	Size	Power3	Power5	Itanium2	Opteron	X1E	ES	SX-8
64	512^3	0.49 (32)	2.31 (30)	1.86 (33)	1.15 (26)	4.49 (25)	3.36 (42)	5.87 (37)
256	512^3	0.45 (30)	2.02 (27)	1.51 (27)	1.08 (25)	4.43 (25)	3.35 (42)	5.86 (37)
512	1024^3	—	2.04 (27)	1.79 (27)	—	4.62 (26)	3.16 (39)	—
1024	1024^3	—	—	1.54 (26)	—	—	3.12 (39)	—

4 CACTUS

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 [18, 8] 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 [7], this study focuses exclusively on the GR solver, which implements the Arnowitt-Deser-Misner (ADM) Baumgarte-Shapiro-Shibata-Nakamura (BSSN) [8] method for stable evolutions of black holes. Figure 2 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.

Fig. 2. Visualization of grazing collision of two black holes as computed by Cactus[1].



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 [8]. The evolution equations can be solved using a number of different numerical approaches, including staggered leapfrog, McCormack, Lax-Wendroff, and iterative Crank-Nicholson schemes. A "lapse" function describes the time slicing between hypersurfaces for each step in the evolution. 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.

4.1 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 AVL 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.

4.2 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.

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

[1] Visualization by Werner Benger (AEI/ZIB) using Amira [19]

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 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 of 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 SX8 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.

The microprocessor based systems offered lower peak performance and generally lower efficiency than the NEC vector systems. The Opteron, however, offered impressive efficiency as well as peak performance in comparison to the Power3 and the Itanium2. Unlike the Power3, the Opteron 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. The Power5 shows much improved performance over the Power3 for the larger problem size thanks to much improved memory bandwidth and through more advanced prefetch features. For the large case, it approaches the efficiency of the Opteron and is the highest performing superscalar system.

Table 5. Cactus performance in GFlop/s (per processor) on 80x80x80 and 250x64x64 grids shown for a range of concurrencies. Percentage of peak is shown in parenthesis.

P	Size	Power3	Power5	Itanium2	Opteron	X1	ES	SX-8
16	80^3	0.31 (21)	1.12 (15)	0.60 (11)	0.98 (22)	0.54 (4)	1.47 (18)	1.86 (12)
64	80^3	0.22 (14)	1.04 (14)	0.58 (10)	0.81 (18)	0.43 (3)	1.36 (17)	1.81 (11)
256	80^3	0.22 (14)	1.12 (15)	0.58 (10)	0.76 (17)	0.41 (3)	1.35 (17)	1.75 (11)
16	$250x64^2$	0.10 (6)	1.07 (14)	0.58 (10)	0.82 (19)	0.81 (6)	2.83 (35)	4.27 (27)
64	$250x64^2$	0.08 (6)	0.95 (13)	0.57 (10)	0.92 (21)	0.72 (6)	2.70 (34)	4.04 (25)
256	$250x64^2$	0.07 (5)	0.95 (13)	0.55 (10)	0.68 (16)	0.68 (5)	2.70 (34)	3.87 (24)

5 Conclusions

This study examined three scientific codes on the parallel vector architectures of the X1/X1E, ES and SX-8, and four superscalar platforms, Power3, Power5, Itanium2, and Opteron. Overall results show that the SX-8 achieved the highest performance of any architecture tested to date, demonstrating the tremendous potential of modern parallel vector systems. However, the SX-8 could not match the computational efficiency of the ES, due in part, to a relatively higher memory latency and higher overhead for irregular data accesses. Both the SX-8 and ES also consistently achieved a significantly higher fraction of peak than the X1/X1E, due to superior scalar processor performance, memory bandwidth, and network bisection bandwidth relative to the peak vector flop rate. Finally, a comparison of the superscalar platforms shows the Power5 having the best absolute performance overall, sometimes overtaking the X1. However, it is often less efficient than the Opteron processor, which consistently outperforms the Itanium2 and Power3 in terms of both raw speed and efficiency. Itanium2 exceeds the performance of the older Power3 processor, however its obtained percentage of peak often falls below that of Power3. Our study also shows that the slide in the sustained performance of microprocessor cores is not irreversible if microprocessor architects are willing to invest the effort to make architectural decisions that eliminate bottlenecks in scientific applications. For instance, the Power5 shows some improvement over its predecessors (the Power3 and Power4) in the execution efficiency for the all the applications, thanks to dramatically improved memory bandwidth and increased attention to latency hiding through advanced prefetch features. Future work will expand our study to include additional areas of computational sciences, while examining the latest generation of supercomputing platforms, including BG/L, X1E, and XT3.

Acknowledgments

The authors would like to thank the staff of the Earth Simulator Center, especially Dr. T. Sato, S. Kitawaki and Y. Tsuda, for their assistance during our visit. We are also grateful for the early SX-8 system access provided by HLRS, Stuttgart, Germany. This research used the resources of several computer centers supported by the Office of Science of the U.S. Department of Energy: the National Energy Research Scientific Computing Center under Contract No. DE-AC02-05CH11231; Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48; the National Center for Computational Sciences at Oak Ridge National Laboratory under Contract No. DE-AC05-00OR22725. The authors were supported by the Office of Advanced Scientific Computing Research in the Department of Energy Office of Science under contract number DE-AC02-05CH11231.

References

1. Russell, R.: The CRAY-1 Computer System. Comm. ACM **V 21, N 1** (1978)

2. Oliker, L., Canning, A., Carter, J., Shalf, J., Ethier, S.: Scientific computations on modern parallel vector systems. In: Proc. SC2004: High Performance Computing, Networking, and Storage Conference. (2004)
3. Oliker, L., et al.: Evaluation of cache-based superscalar and cacheless vector architectures for scientific computations. In: Proc. SC2003: High Performance Computing, Networking, and Storage Conference. (2003)
4. Dongarra, J., Luszczek, P.: HPC Challenge Benchmarks - Stream EP. (http://icl.cs.utk.edu/hpcc/index.html)
5. Carter, J., Soe, M., Oliker, L., Tsuda, Y., Vahala, G., Vahala, L., Macnab, A.: Magnetohydrodynamic turbulence simulations on the Earth Simulator using the lattice Boltzmann method. In: Proc. SC2005: High performance computing, networking, and storage conference. (2005)
6. Vahala, G., Yepez, J., Vahala, L., Soe, M., Carter, J.: 3D entropic lattice Boltzmann simulations of 3D Navier-Stokes turbulence. In: Proc. of 47th Annual Meeting of the APS Division of Plasma Physics. (2005)
7. Font, J.A., Miller, M., Suen, W.M., Tobias, M.: Three dimensional numerical general relativistic hydrodynamics: Formulations, methods, and code tests. Phys. Rev. D **61** (2000)
8. Alcubierre, M., Allen, G., Brgmann, B., Seidel, E., Suen, W.M.: Towards an understanding of the stability properties of the 3+1 evolution equations in general relativity. Phys. Rev. D **(gr-qc/9908079)** (2000)
9. Succi, S.: The lattice Boltzmann equation for fluids and beyond. Oxford Science Publ. (2001)
10. Dellar, P.: Lattice kinetic schemes for magnetohydrodynamics. J. Comput. Phys. **79** (2002)
11. Macnab, A., Vahala, G., Pavlo, P., , Vahala, L., Soe, M.: Lattice Boltzmann model for dissipative incompressible MHD. In: Proc. 28th EPS Conference on Controlled Fusion and Plasma Physics. Volume 25A. (2001)
12. Macnab, A., Vahala, G., Vahala, L., Pavlo, P.: Lattice Boltzmann model for dissipative MHD. In: Proc. 29th EPS Conference on Controlled Fusion and Plasma Physics. Volume 26B., Montreux, Switzerland (June 17-21, 2002)
13. Ansumali, S., Karlin, I.V.: Stabilization of the lattice Boltzmann method by the H theorem: A numerical test. Phys. Rev. **E62** (2000) 7999–8003
14. Ansumali, S., Karlin, I.V., Öttinger, H.C.: Minimal entropic kinetic models for hydrodynamics. Europhys. Lett. **63 (6)** (2003)
15. Wellein, G., Zeiser, T., Donath, S., Hager, G.: On the single processor performance of simple lattice bolzmann kernels. Computers and Fluids **Accepted for Publication, http://dx.doi.org/10.1016/j.compfluid.2005.02.008** (2005)
16. Worley, P.: (Private communication)
17. Carter, J., Oliker, L.: Performance evaluation of lattice-Boltzmann magnetohyddrodynamics simulations on modern parallel vector systems. In: Proc. of the 2nd Teraflop Workshop, Lecture Notes in Computer Science (LNCS). (2006)
18. Schnetter, E., et al.: Cactus Code Server. (http://www.cactuscode.org)
19. TGS Inc.: Amira - Advanced 3D Visualization and Volume Modeling. (http://www.amiravis.com)