Performance Evaluation of Lattice-Boltzmann Magnetohydrodynamics Simulations on Modern Parallel Vector Systems

Jonathan Carter and Leonid Oliker

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

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 such platforms has become major concern in high performance computing. The latest generation of custom-built parallel vector systems have the potential to address this concern for numerical algorithms with sufficient regularity in their computational structure. In this work, we explore two and three dimensional implementations of a lattice-Boltzmann magnetohydrodynamics (MHD) physics application, on some of today's most powerful supercomputing platforms. Results compare performance between the the vector-based Cray X1, Earth Simulator, and newly-released NEC SX-8, with the commodity-based superscalar platforms of the IBM Power3, Intel Itanium2, and AMD Opteron. Overall results show that the SX-8 attains unprecedented aggregate performance across our evaluated applications.

1 Introduction

The last decade has witnessed a rapid proliferation of superscalar cache-based microprocessors to build high-end computing (HEC) platforms. This is primarily because their generality, scalability, and cost effectiveness convinced computer vendors and users that vector architectures hold little promise for future large-scale supercomputing systems. However, the constant degradation of superscalar sustained performance has become a well-known problem in the scientific computing community. This trend has been widely attributed to the use of superscalar-based commodity components 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 are addressing these challenges for numerical algorithms amenable to vectorization.

Superscalar architectures are unable to efficiently exploit the large number of floating-point units that can be potentially fabricated on a chip, due to the small granularity of their instructions and the correspondingly complex control structure necessary to support it. Vector technology, on the other hand, provides an

efficient approach for controlling a large amount of computational resources provided that sufficient regularity in the computational structure can be discovered. Vectors exploit these regularities to expedite uniform operations on independent data elements, 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, thus reducing control complexity and efficiently controlling a large amount of computational resources. However, when such operational parallelism cannot be found, the efficiency of the vector architecture can suffer from the properties of Amdahl's Law, where the time taken by the portions of the code that are non-vectorizable easily dominate the execution time.

In order to quantify what modern vector capabilities entail for the scientific communities that rely on modeling and simulation, it is critical to evaluate them in the context of demanding computational algorithms. This work compares performance between the vector-based Cray X1, Earth Simulator (ES) and newly-released NEC SX-8, with commodity-based superscalar platforms: the IBM Power3, Intel Itanium2, and AMD Opteron. We study the behavior of two scientific codes with the potential to run at ultra-scale, in the areas of MHD physics simulations (LBMHD2D and LBMHD3D). Our work builds on our previous efforts [1,2] 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 for LBMHD2D. 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.

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 five parallel HEC architectures. Observe that the vector machines have higher peak performance and better system balance than the superscalar platforms. Additionally, the X1, ES, and SX-8 have high memory bandwidth 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.

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 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 AMD Opteron system, Jacquard, 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 proces-

sors (4.4 Gflop/s peak), interconnected via Infiniband fabric in a fat-tree configuration. Finally, the Intel Itanium experiments were performed on the Thunder system, consisting 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, and is located at Lawrence Livermore National Laboratory.

We also examine three state-of-the-art parallel vector systems. The Cray X1 is designed to combine traditional vector strengths with the generality and scalability features of modern superscalar cache-based parallel systems. The computational core, called the single-streaming processor (SSP), contains two 32-stage vector pipes running at 800 MHz. Each SSP contains 32 vector registers holding 64 double-precision words, 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 SSP can be combined into a logical computational unit called the multi-streaming processor (MSP) with a peak of 12.8 Gflop/s. 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, and large system configuration are networked through a modified 2D torus interconnect. All reported X1 experiments were performed on the 512-MSP system (several reserved for system services) running UNICOS/mp 2.5.33 (5.3 programming environment) and operated by Oak Ridge National Laboratory.

The vector processor of the ES uses a dramatically different architectural approach than conventional cache-based systems. Vectorization exploits regularities in the computational structure of scientific applications to expedite uniform operations on independent data sets. The 500 MHz ES processor is an enhanced NEC SX6, containing an 8-way replicated vector pipe with a peak performance of 8.0 Gflop/s per CPU. The Earth Simulator is the world's third most powerful supercomputer [3], containing 640 ES nodes connected through a custom single-stage IN crossbar. 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

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

Platform	CPU/ Node	Clock (MHz)	Peak (GF/s)	Mem BW (GB/s)	Peak (Byte/Flop)
Power3	16	375	1.5	0.7	0.47
Itanium2	4	1400	5.6	6.4	1.1
Opteron	2	2200	4.4	6.4	1.5
X1	4	800	12.8	34.1	2.7
ES (Modified SX-6)	8	500	8.0	32.0	4.0
SX-8	8	2000	16.0	64.0	4.0

during the authors' visit to the Earth Simulator Center located in Kanazawa-ku, Yokohama, Japan in 2003 and 2004.

Finally, we examine the newly-released 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 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 36 node (72 soon to be 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).

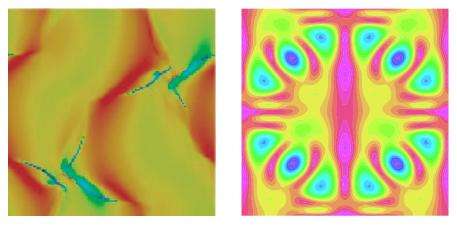## 3   Magneto-Hydrodynamic Turbulence Simulation

Lattice Boltzmann methods (LBM) have proved a good alternative to conventional numerical approaches for simulating fluid flows and modeling physics in fluids [4]. 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) [5, 6] with promising results. We use two LB MHD codes, a previously used 2D code [1, 7] and a more recently developed 3D code. In both cases, the codes simulate the behavior of a conducting fluid evolving from simple initial conditions through the onset of turbulence. Figure 1 shows a slice through the xy-plane in the (left) 2D and right (3D) simulation, where the vorticity profile has considerably distorted after several hundred time steps as computed by LBMHD. In the 2D case, the square spatial grid is coupled

Table 2. Interconnect performance of the Power3, 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
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 (Modified SX-6)	Custom (IN)	5.6	1.5	0.19	Crossbar
SX-8	IXS	5.0	2.0	0.13	Crossbar

to an octagonal streaming lattice and block distributed over a 2D processor grid as shown in Figure 2. The 3D spatial grid is coupled via 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 proportional to the number of streaming directions — in this case 9 and 27 (8 and 26 plus the null vector).

Fig. 1. Contour plot of xy-plane showing the evolution of vorticity from well-defined tube-like structures into turbulent structures using (left) LBMHD2D and (right) LBMHD3D.

The simulation proceeds by a sequence of collision and stream steps. A collision step involves data local only to that spatial point, allowing concurrent, dependence-free point updates; the mesoscopic variables at each point are updated through a complex algebraic expression originally derived from appropriate conservation laws. A stream step evolves the mesoscopic variables along the streaming lattice, necessitating communication between processors for grid points at the boundaries of the blocks.

Additionally, for the 2D case, an interpolation step is required between the spatial and streaming lattices since they do not match. This interpolation is folded into the stream step. For the 3D case, a key optimization described by Wellein and co-workers [8] 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.
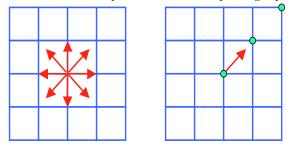
Fig. 2. Octagonal streaming lattice superimposed over a square spatial grid (left) requires diagonal velocities to be interpolated onto three spatial gridpoints (right).

3.1 Vectorization details

The basic computational structure consists of two or three nested loops over spatial grid points (typically 1000s iterations) with inner loops over velocity streaming vectors and magnetic field streaming vectors (typically 10-30 iterations), performing various algebraic expressions. Although the two codes have kernels which are quite similar, our experiences in optimizing were somewhat different.

For the 2D case, in our earlier work on the ES, attempts to make the compiler vectorize the inner gridpoint loops rather than the streaming loops failed. The inner grid point loop was manually taken inside the streaming loops, which were hand unrolled twice in the case of small loop bodies. In addition, the array temporaries added were padded to reduce bank conflicts. With the hindsight of our later 3D code experience, this strategy is clearly not optimal. Better utilization of the multiple vector pipes can be achieved by completely unrolling the streaming loops and thus increasing the volume of work within the vectorized loops. We have verified that this strategy does indeed give better performance than the original algorithm on both the ES and SX-8, and show results that illustrate this in the next section. Turning to the X1, the compiler did an excellent job, multi-streaming the outer grid point loop and vectorizing the inner grid point loop after unrolling the stream loops without any user code restructuring. For the superscalar architectures some effort was made to tune for better cache use. First, the inner gridpoint loop was blocked and inserted into the streaming loops to provide stride-one access in the innermost loops. The streaming loops were then partially unrolled.

For the 3D 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 the X1. In the case of the X1, 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. The difference in behavior is clearly related to the size of the unrolled loop body, the 3D case being a factor of

approximately three more complicated. In the case of X1 the number of vector registers available for a vectorized loop is more limited than for the SX systems and for complex loop bodies register spilling will occur. However, in this case, the strategy pays off as shown experimental results section below. For the superscalar architectures, we utilized a data layout that has been previously shown to be optimal on cache-based machines [8], but did not explicitly tune for the cache size on any machine.

Interprocessor communication was implemented using the MPI library, by copying the non-contiguous mesoscopic variables data into temporary buffers, thereby reducing the required number of send/receive messages.

These codes represent candidate ultra-scale applications that have the potential to fully utilize leadership-class computing systems. Performance results, presented in Gflop/s per processor and percentage of peak, are used to compare the relative time to solution of our evaluated computing systems. Since different algorithmic approaches are used for the vector and scalar implementations, this value is computed by dividing a baseline flop-count (as measured on the ES) by the measured wall-clock time of each platform.

3.2   Experimental Results

Tables 3 and 4 present the performance of both LBMHD applications across the five architectures evaluated in our study. Cases where the memory required exceeded that available are indicated with a dash. For LBMHD2D we show the performance of both vector algorithms (first strip-mined as used in the original ES experiment, and second the new unrolled inner) for the SX-8. In accordance with the discussion in the previous section, the new algorithm clearly outperforms the old.

Table 3. LBMHD2D performance in GFlop/s (per processor) across the studied architectures for a range of concurrencies and grid sizes. The original and optimized algorithms are shown for the ES and SX-8. Percentage of peak is shown in parenthesis.

P	Size	Power3	Itanium2	Opteron	X1	original ES	optimized ES	original SX-8	optimized SX-8
16	4096^2	0.11 (7)	0.40 (7)	0.83 (19)	4.32 (34)	4.62 (58)	5.00 (63)	6.33 (40)	7.45 (47)
64	4096^2	0.14 (9)	0.42 (7)	0.81 (18)	4.35 (34)	4.29 (54)	4.36 (55)	4.75 (30)	6.28 (39)
64	8192^2	0.11 (7)	0.40 (7)	0.81 (18)	4.48 (35)	4.64 (58)	5.01 (62)	6.01 (38)	7.03 (44)
256	8192^2	0.12 (8)	0.38 (6)		2.70 (21)	4.26 (53)	4.43 (55)	4.44 (28)	5.51 (34)

Observe that the vector architectures clearly outperform the scalar systems by a significant factor. Across these architectures, the LB applications exhibit an average vector length (AVL) very close to the maximum and a very high vector operation ratio (VOR). In absolute terms, the SX-8 is the leader by a wide

Table 4. 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	Itanium2	Opteron	X1	ES	SX-8
16	256^3	0.14 (9)	0.26 (5)	0.70 (16)	5.19 (41)	5.50 (69)	7.89 (49)
64	256^3	0.15 (10)	0.35 (6)	0.68 (15)	5.24 (41)	5.25 (66)	8.10 (51)
256	512^3	0.14 (9)	0.32 (6)	0.60 (14)	5.26 (41)	5.45 (68)	9.66 (60)
512	512^3	0.14 (9)	0.35 (6)	0.59 (13)	—	5.21 (65)	—

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 X1 behavior, we see that in MSP mode 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.

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 2-way SMP Opteron node has STREAM memory bandwidth [9] of more than twice that of the Itanium2 [10], which utilizes a 4-way SMP node configuration. 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.

Although the SX-8 achieves the highest absolute performance, the percentage of peak is somewhat lower than that of ES. We believe that this is related to the memory subsystem and use of commodity DDR-SDRAM. In order to test this hypothesis, we recorded the time due to memory bank conflicts for both applications on the ES and SX-8 using the ftrace tool, and present it in Table 5.

Most obviously in the case of the 2D code, the amount of time spent due to bank conflicts is appreciably larger for the SX-8. Efforts to reduce the amount of time for bank conflicts for the 2D 64 processor benchmark produced a slight improvement to 13%. In the case of the 3D code, the effects of bank conflicts are minimal.

4 Conclusions

This study examined two scientific codes on the parallel vector architectures of the X1, ES and SX-8, and three superscalar platforms, Power3, Itanium2, and

Table 5. LBMHD2D and LBMHD3D bank conflict time (as percentage of real time) shown for a range of concurrencies and grid sizes on ES and SX-8.

$Code$	P	Grid Size	ES BC (%)	SX-8 BC (%)
2D	64	8192^2	0.3	16.6
2D	256	8192^2	0.3	10.7
3D	64	256^3	>0.01	1.1
3D	256	512^3	>0.01	1.2

Opteron. A summary of the results for the largest comparable problem size and concurrency is shown in Figure 3, for both (left) raw performance and (right) percentages of peak. 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 sustained performance of the ES, due in part, to a relatively higher memory latency overhead for irregular data accesses. Both the SX-8 and ES also consistently achieved a significantly higher fraction of peak than the X1, 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 that the Opteron consistently outperformed the Itanium2 and Power3, both in terms of raw speed and efficiency - due, in part, to its on-chip memory controller and (unlike the Itanium2) the ability to store floating point data in the L1 cache. The Itanium2 exceeds the performance of the (relatively old) Power3 processor, however its obtained percentage of peak falls further behind. 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.
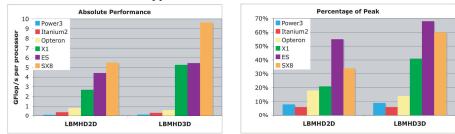
Fig. 3. Summary comparison of (left) raw performance and (right) percentage of peak across our set of evaluated applications and architectures.

Acknowledgments

References

1. 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)
2. 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)
3. Meuer, H., Strohmaier, E., Dongarra, J., Simon, H.: Top500 Supercomputer Sites. (http://www.top500.org)
4. Succi, S.: The lattice boltzmann equation for fluids and beyond. Oxford Science Publ. (2001)
5. Dellar, P.: Lattice kinetic schemes for magnetohydrodynamics. J. Comput. Phys. **79** (2002)
6. 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)
7. 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)
8. Wellein, G., Zeiser, T., Donath, S., Hager, G.: On the single processor performance of simple lattice bolzmann kernels. Computers and Fluids (To appear)
9. McCalpin, J.: STREAM benchmark. (http://www.cs.virginia.edu/stream/ref.html)
10. Dongarra, J., Luszczek, P.: HPC challenge benchmark. (http://icl.cs.utk.edu/hpcc/index.html)