Magnetohydrodynamic Turbulence Simulations on the Earth Simulator Using the Lattice Boltzmann Method

Jonathan Carter
NERSC
Lawrence Berkeley National Laboratory, Berkeley, CA 94720 USA
jtcarter@lbl.gov

Min Soe
Department of Mathematics and Sciences
Rogers State University, OK, 74017 USA
msoe@rsu.edu

Leonid Oliker
Computational Research Division
Lawrence Berkeley National Laboratory, Berkeley, CA 94720 USA
loliker@lbl.gov

Yoshinori Tsuda
Earth Simulator Center
Japan Agency for Marine-Earth Science and Technology, Yokohama 236-0001, Japan
tsuda@es.jamstec.go.jp

George Vahala
Department of Physics
College of William & Mary, Williamsburg, VA 23187, USA
vahala@niv.physics.wm.edu

Linda Vahala
Department of Electrical and Computer Engineering
Old Dominion University, Norfolk, VA 23529 USA
lvahala@odu.edu

Angus Macnab
CSCAMM
University of Maryland, College Park, MD 20742 USA
angus@cscamm.umd.edu

ABSTRACT

Highly optimized large-scale lattice Boltzmann simulations of 3D magnetohydrodynamic turbulence are performed on the Earth Simulator. We discuss code optimization schemes for both single processor and parallel performance, and present performance data at various concurrencies and grid sizes. A production run on a 1440^3 grid using 4800 processors achieved a total aggregate performance of over 26 Tflop/s, making this study one of the largest yet undertaken and allowing access to an unprecedented level of detail. While a full analysis will require much more work, representative features of some 3D MHD turbulence are presented.

SC|05 November 12-18, 2005, Seattle, Washington, USA

1. INTRODUCTION

Magnetohydrodynamics (or MHD) describes self-consistently the macroscopic behavior of an electrically conducting fluid by combining the Navier-Stokes equations with Maxwell's equations. MHD turbulence plays an important role in many branches of physics [1]: from astrophysical phenomena in stars, accretion discs, interstellar and intergalactic media to plasma instabilities in magnetic fusion devices. It is well known that the simulation of turbulent flows in complex geometries places great strain on computational algorithms designed for the direct solution of the MHD equations. Sophisticated schemes must be developed to handle the singular matrices that crop up in accurately resolving the nonlinear convective derivatives, e.g., high order finite elements or Newton-Krylov algorithms.

Lattice Boltzmann (LB) schemes are an alternate approach that circumvent the resolution problems with the macroscopic nonlinear convective derivatives by embedding into a higher dimensional (kinetic) phase space. While this appears to be an inverse-Statistical mechanical approach, the resulting kinetic equations can be discretized on a phase space lattice that has a minimal number of (discrete) velocities sufficient that the long-time, long-wavelength

(Chapman-Enskog) limit reproduces the desired macroscopic nonlinear equations. With simple linear advective terms the difficult macroscopic non-local nonlinearities are recovered by simple polynomial (local) nonlinearities in the collision operator of the kinetic equation.

For Navier-Stokes turbulence, one needs to introduce only a scalar distribution whose discrete moments yield the fluid density and mean velocity. While this LB algorithm has been used extensively over the past ten years for simulating Navier-Stokes flows [2], its application to MHD has not been as vigorously pursued—presumably because of the difficulty of introducing the magnetic field at a kinetic level. The first attempts introduced a complex double velocity lattice-streaming algorithm on a scalar distribution function. Recently for 2D MHD, Dellar [3] introduced a separate vector distribution function for the magnetic field whose zero (vector discrete) moment yielded the magnetic field. These sets of coupled kinetic equations could then be discretized on the standard lattices used in fluid turbulence. Here we extend this algorithm to 3D MHD.

In earlier work [4], we noted that the Earth Simulator showed very impressive performance for a variety of scientific applications, including a two-dimensional LB application. The very high memory bandwidth coupled with the large amount of data parallelism present in the LB scheme made this architecture an obvious choice for some of the largest LB MHD simulations attempted so far.

2. Resistive Magnetohydrodynamic Turbulence

Here we consider the compressible resistive MHD equations for the density ρ, velocity u and magnetic field **B**

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}+\mathbf{u}.\nabla\mathbf{u}\right) = -\nabla p + (\nabla\times\mathbf{B})\times\mathbf{B} + \mu\nabla^2\mathbf{u}$$
$$\frac{\partial \mathbf{B}}{\partial t} = \nabla\times(\mathbf{u}\times\mathbf{B}) + \eta\nabla^2\mathbf{B} \qquad (1)$$
$$\frac{\partial \rho}{\partial t} + \nabla.(\rho\,\mathbf{u}) = 0$$

Closure is achieved by an isothermal equation of state: $p = \rho\, c_s^2$ with constant sound speed c_s and $\nabla.\,\mathbf{B} = 0$.

The disparate length and time scales that appear in the solutions of resistive MHD require careful numerical treatment. A semi-implicit time scheme with high-order finite element spatial discretization will result in the inversion of ill-conditioned matrices which appear as a manifestation of the underlying numerical instabilities inherent in the explicit numerical scheme. The ill-conditioned matrices are handled in Jacobi-free Newton Krylov techniques by suitably chosen pre-conditioners. While the implicit time step interval can be on the order of 200 times greater than that permitted by Courant-Friedrichs-Levy (CFL) constraints on standard explicit codes, the overall speed-up of the implicit codes is only a factor of 10-30 faster due to all the extra computational effort.

Here we present an alternative explicit scheme[3, 5] to direct MHD solvers, which bypasses the difficult resolution of the nonlinear convective derivatives by embedding the problem into a higher dimensional kinetic phase space. In this lattice Boltzmann method (LBM), we discretize a set of coupled linear scalar-vector Bhatnagar-Gross-Krook (BGK)[6] kinetic equations

$$\frac{\partial f(\mathbf{x},\boldsymbol{\xi},t)}{\partial t} + \boldsymbol{\xi}\mathbf{g}\nabla f(\mathbf{x},\boldsymbol{\xi},t) = -\frac{1}{\tau_u}\left[f(\mathbf{x},\boldsymbol{\xi},t) - f^{eq}(\mathbf{x},\boldsymbol{\xi},t)\right]$$
$$\frac{\partial \mathbf{g}(\mathbf{x},\boldsymbol{\xi},t)}{\partial t} + \boldsymbol{\xi}\mathbf{g}\nabla \mathbf{g}(\mathbf{x},\boldsymbol{\xi},t) = -\frac{1}{\tau_m}\left[\mathbf{g}(\mathbf{x},\boldsymbol{\xi},t) - \mathbf{g}^{eq}(\mathbf{x},\boldsymbol{\xi},t)\right] \qquad (2)$$

where one connects back to the macroscopic variables by the appropriate moments:

$$\rho(\mathbf{x},t) = \int d\boldsymbol{\xi}\; f(\mathbf{x},\boldsymbol{\xi},t)$$
$$\rho\,\mathbf{u}(\mathbf{x},t) = \int d\boldsymbol{\xi}\; f(\mathbf{x},\boldsymbol{\xi},t)\,\boldsymbol{\xi} \qquad (3)$$
$$\mathbf{B}(\mathbf{x},t) = \int d\boldsymbol{\xi}\; \mathbf{g}(\mathbf{x},\boldsymbol{\xi},t)$$

τ_u and τ_m are the relaxation rates at which collisions drive the distribution functions to their equilibrium states: $f \rightarrow f^{eq}$ and $\mathbf{g} \rightarrow \mathbf{g}^{eq}$. The nonlinear convective derivatives in MHD are now replaced by linear kinetic advective terms in the LBM. On discretizing the kinetic equations, one chooses the velocity lattice geometries and the polynomial representation of $f^{eq}(\mathbf{u},\mathbf{B})$ and $\mathbf{g}^{eq}(\mathbf{u},\mathbf{B})$ so that in the Chapman-Enskog (long wavelength, long time) limit one recovers the original nonlinear resistive MHD equations (1).

For 3D resistive MHD, an appropriate phase velocity lattice has 27 velocities: $\boldsymbol{\xi} \rightarrow \mathbf{c}_\alpha$, α = 1,…,27, where the speeds on a unit cube are √2 [velocities of the form (±1,±1,0), α = 1,…,12], 1 [velocities of the form (±1,0,0), α = 13,…,18], √3 [velocities of the form (±1,±1,±1), α = 19,…,26] as well as one rest particle α = 27. The lattice symmetries are such that the resulting discretized kinetic equations ($\Delta x = 1 = \Delta t$)

$$f_\alpha(\mathbf{x}+\mathbf{c}_\alpha,t+1) = f_\alpha(x,t) - \frac{1}{\tau_u}\left[f_\alpha(\mathbf{x},t) - f_\alpha^{eq}(x,t)\right], \quad \alpha = 1..... 27$$
$$\mathbf{g}_\alpha(\mathbf{x}+\mathbf{c}_\alpha,t+1) = \mathbf{g}_\alpha(\mathbf{x},t) - \frac{1}{\tau_m}\left[\mathbf{g}_\alpha(\mathbf{x},t) - \mathbf{g}_\alpha^{eq}(\mathbf{x},t)\right], \quad \alpha = 13, \ldots, 27 \qquad (4)$$

are fully explicit, second order in space and time. The asymmetry in the choice of the (scalar) velocity and (vector) magnetic distributions arises from the different symmetry properties of the second moment tensors arising in Eq. (2):

$$\Pi_{ij} = \left(\rho c_s^2 + \frac{\mathbf{B}^2}{2} \right) \delta_{ij} + \rho u_i u_j - B_i B_j = \Pi_{ji} = \sum_{\alpha=1}^{27} f_\alpha c_{\alpha i} c_{\alpha j}$$

$$\Lambda_{ij} = B_i u_j - B_j u_i = -\Lambda_{ji} = \sum_{\alpha=13}^{27} c_{\alpha i}\, g_{\alpha j} \qquad (5)$$

Moreover, because closure for the $\mathbf{g}_\alpha$-equation is attained at the 1st moment (while that for f_α-equation is attained at the 2nd moment), the number of phase space velocities to recover information on the magnetic field is reduced from 27 to 15. The nonlocal nonlinearities in the resistive MHD are recovered from the linear LBM through the quadratic local field nonlinearities in the relaxation distribution functions:

$$f_\alpha^{eq} = f_\alpha^{eq}\left(\mathbf{u}\cdot\mathbf{c}_\alpha,\, \mathbf{B}\cdot\mathbf{c}_\alpha,\, \mathbf{u}^2,\, \mathbf{B}^2\right),$$
$$\mathbf{g}_\alpha^{eq} = \mathbf{g}_\alpha^{eq}\left(\mathbf{B},\, \mathbf{B}\cdot\mathbf{c}_\alpha,\, \mathbf{u}\cdot\mathbf{c}_\alpha\right) \qquad (6)$$

and the transport coefficients are related to relaxation rates

$$\mu = \frac{1}{3}\left(\tau_u - \frac{1}{2}\right), \qquad \eta = \frac{1}{3}\left(\tau_m - \frac{1}{2}\right) \qquad (7)$$

Thus, in a minimal dimensional discrete phase space the evolution of the distribution functions in LBM are obtained from: (a) BGK collisional relaxation, which uses only local information at each spatial node, and (b) Lagrangian streaming of this information from one spatial node to neighboring nodes. This makes LBM ideal for large-scale vector parallel machines like the Earth Simulator.

As can be expected from explicit algorithms, but with kinetic CFL=1, LBM is prone to numerical nonlinear instabilities as one pushes to even higher Reynolds numbers. These numerical instabilities arise since there are no constraints imposed to enforce the distribution functions to remain non-negative. Such entropic LBM algorithms, which do persevere the non-negativity of the distribution functions---even in the limit of arbitrary small transport coefficients---now do exist for Navier-Stokes turbulence [7], and there is active research in developing such entropic LBM algorithms for MHD.

Finally, we should comment on the important constraint $\nabla\cdot\mathbf{B} = 0$ [8]. While LBM does not enforce this constraint explicitly, numerous tests have indicated that the algorithm somehow keeps $\nabla\cdot\mathbf{B}$ within acceptable bounds without the need for divergence cleaning.

3. The Earth Simulator

The Earth Simulator (ES) hardly needs an introduction. Since it debuted in the spring of 2002 it has attracted worldwide attention─it occupied the number 1 spot on the Top 500[9] list for a record two and a half years before moving down the list in November 2004.

The ES uses a dramatically different architectural approach than conventional cache-based systems that comprise the rest of the top 10 spots in the Top 500 list. Powerful vector processors are connected via a fast single stage switch.

The 1000 MHz ES processor contains an 4-way replicated vector pipe capable of issuing a multiply-add each cycle, for a peak performance of 8.0 Gflop/s per CPU. The processors contain 72 vector registers, each holding 256 64-bit words (vector length = 256). For non-vectorizable instructions, the ES contains a 500 MHz scalar processor with a 64 KB instruction cache, a 64 KB data cache, and 128 general-purpose registers. The 4-way superscalar unit has a peak of 1.0 Gflop/s and supports branch prediction, data prefetching, and out-of-order execution.

Like traditional vector architectures, the ES vector unit is cacheless. Memory latencies are masked by overlapping pipelined vector operations with memory operations. The main memory chip for the ES uses a specially developed high speed DRAM called FPLRAM (Full Pipelined RAM) operating at 24 ns bank cycle time. Each SMP contains eight processors that share the nodes memory with a bidirectional bandwidth of 32 GB/s.

The ES contains 640 nodes connected through a custom single-stage crossbar. This high-bandwidth interconnect topology provides impressive communication characteristics, as all nodes are a single hop from one another. The peak performance of the interconnect is 12.3 GB/s (11.8 GB/s measured with MPI) in each direction. The latency of most internode MPI functions is approximately 6μs [10].

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 results reported here were performed during the authors' visit to the Earth Simulator Center located in Kanazawa-ku, Yokohama, Japan in October 2004, and later by one of us (YT) with local access.

4. Computational Implementation

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 MHD simulations under periodic boundary conditions, with the spatial grid and phase space velocity lattice overlaying each other. Thus, the time-advanced streamed distribution functions lie directly on the spatial lattice nodes and no interpolation is needed. Of course, interpolation would be needed if non-uniform spatial grids had been introduced. The structure of the program is straightforward, two multi-dimensional arrays hold the particle distribution function, *f,* and magnetic field distribution function, **g**, for each of the rectilinear mesh of grid points. We use the 3DQ27 model which, as previously noted, uses 27 discrete velocities for *f*, and 15 (corresponding to the last 15 vectors of *f*) for **g**. The

arrays are further doubled in size to accommodate the values for the current and next time step in the simulation. Approximately 1 KB of storage is required per grid point, leading to quite large memory requirements for relatively small grids.

After initialization, the simulation proceeds conceptually via two phases, each repeated for every time step. In the first phase, *collision*, the macroscopic quantities, density, momentum density, and magnetic field are constructed from the moment expressions detailed above, the equilibrium values f^{eq} and $\mathbf{g}^{eq}$ calculated, and updated values of f and $\mathbf{g}$ are calculated from (4). In the second phase, *stream*, the updated values are streamed to the appropriate neighboring cell according to the value of $\mathbf{c}_{\alpha}$. The first step is computationally intensive, but requires only data local to the grid point. The second step is a set of shift operations, moving data from grid point to grid point according to the lattice vector.

Wellein and co-workers have discussed optimal layouts for the particle distribution function functions in the case of LB fluid dynamics [11]. For most architectures they found the "propagation optimized layout" to be optimal, where the first dimensions are the Cartesian coordinates, followed by an index representing the streaming vector, i.e. `f(x,y,z,27)`. We have followed this choice, and simply extended it for the magnetic field distribution function, `g(x,y,z,13:27,3)`. Fortran array syntax is assumed with x varying fastest when stepping contiguously through memory. The simplest implementation is outlined in the code fragments shown in Figures 1 and 2.

```
function collision
dimension f(0:nx+1,0:ny+1,0:nz+1,27),
   g(0:nx+1,0:ny+1,0:nz+1,13:27,3)
dimension feq(0:nx+1,0:ny+1,0:nz+1,27),
   geq(0:nx+1,0:ny+1,0:nz+1,13:27,3)

do x=1,nx: do y=1,ny: do z=1,nz
 do i=1,27
! compute density, momentum density
   density+=f(x,y,z,i)
   …
 end do
 do i=13,27
! compute magnetic field
   b(1)+=g(x,y,z,i,1)
   …
 end do

 do i=1,27
! compute feq
   feq(x,y,z,i)=…
   …
 end do
 do i=13,27
! compute geq
   geq(x,y,z,i,1)=…
   …
 end do
end do: endo do: end do
```

Figure 1 Outline of initial collision routine

For the parallel implementation each array 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. For ghost cell updates during the stream phase, we use the shift algorithm [12]. In this method we make use of the fact that after the first exchange is completed in one direction, we have partially populated ghost cells. The

```
function stream
dimension f(0:nx+1,0:ny+1,0:nz+1,27),
   g(0:nx+1,0:ny+1,0:nz+1,13:27,3)
dimension feq(0:nx+1,0:ny+1,0:nz+1,27),
   geq(0:nx+1,0:ny+1,0:nz+1,13:27,3)

! update all ghost cells, only 1 strip
! is shown
do dir={1,2,9,11,13,14,16,18,24}
  feq(nx,:,:,dir) sent via MPI calls
  feq(0,:,:,dir) recv via MPI calls
end do
do dir={13,14,16,18,24}
  geq(nx,:,:,dir,:) sent via MPI calls
  geq(0,:,:,dir,:) recv via MPI calls
end do
…

do x=1,nx: do y=1,ny: do z=1,nz
! stream feq and geq values to appropriate
! neighboring cells
  f(x,y,z,1)=feq(x+1,y,z,1)
  f(x,y,z,2)=feq(x,y+1,z,2)
…
  g(x,y,z,1,:)=geq(x+1,y,z,1,:)
  g(x,y,z,2,:)=geq(x,y+1,z,2,:)

end do: endo do: end do
```

Figure 2 Outline of initial stream routine

next exchange includes this data, further populating the ghost cells. A diagram showing the 2D case is shown in Figure 3.

This procedure has the advantage of reducing the number of neighbors included in message-passing from 26 to 6, a beneficial optimization considering the MPI latency of the ES is reasonably high compared with the bandwidth. Because different lattice vectors contribute to different spatial directions, the data to be exchanged are not contiguous. For example, 12 of the 26 lattice vectors have a component in the +x direction, and must be sent in this direction, but are not contiguous in the arrays `f` or `g`. The data is packed into a single buffer, resulting in 6 message exchanges per time step. The initial implementation made use of `mpi_isend`/`mpi_irecv` pairs.

Initial experiments using 4 processors on the ES showed very poor single processor performance, about 330 Mflop/s. Using the `ftrace` tool showed that while the vector operation ratio (VOR) was reasonably high at 89%, the average vector length (AVL) was very short at around 10. Inspection of the compiler listing showed that

the innermost loops (see Figure 1) had been vectorized, corroborating the `ftrace` output. To improve performance, the innermost loops were unrolled using compiler directives and the innermost grid point loop (over coordinate z) vectorized giving a much-improved result of 3.97 Gflop/s with an AVL of over 255 and VOR of over 99%. We should note that the choice of mapping lattice to processor grid sets the dimensions of nx, ny, and nz. This in turn affects the overall vector length in the collision routine. In all our experiments the decomposition was chosen to preserve a vector length of at least 256. The decompositions used are shown in Table 1.

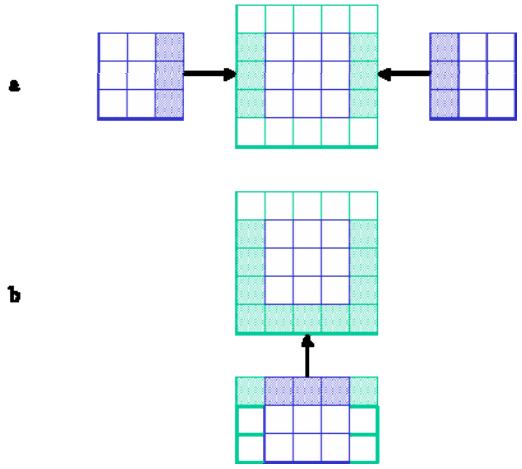


Figure 3 a) ghost cells (green) populated by first exchange b) this data then used to populate 'corner' ghost cells when exchanged in other direction

A second key optimization, described by several groups[13, 11], was then implemented. They noticed that the two phases of the LB 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.

The memory access pattern for the collision phase becomes much more complex, but the amount of data transferred each time step is reduced dramatically. Figure 4 attempts to sketch how we implemented the gather version of this algorithm. The stream function corresponding to this new collision contains only the code to exchange ghost cell values via MPI.

We benchmarked the effect of this algorithm change on a small 256^3 grid simulation running on 16 processors. The effect was to boost the per processor performance by 13% to 4.8 Gflop/s, and produce a decrease in time to solution of 12%.

Turning to parallel performance, we examined the behavior of 256 and 512 processor simulations for an intermediate 512^3 grid. A speedup of 1.978 was obtained on doubling the processor count, with the larger run achieving 5.2 Gflop/s per processor. Two experiments to improve MPI performance were conducted, to see if the already outstanding scaling could be improved. The first was to replace the `mpi_isend/ mpi_irecv/ mpi_wait` calls with simple `mpi_sendrecv` calls. This was motivated by the fact that little computational work (just a small amount of copying to the send buffers) could be overlapped with communication, and that the `mpi_wait` calls would likely be associated with some

```
function collision
dimension f(0:nx+1,0:ny+1,0:nz+1,27),
g(0:nx+1,0:ny+1,0:nz+1,13:27,3)
dimension feq(0:nx+1,0:ny+1,0:nz+1,27),
geq(0:nx+1,0:ny+1,0:nz+1,13:27,3)
dimension ft(27), gt(27,3)

do x=1,nx: do y=1,ny: do z=1,nz
! collect feq and geq values from
! appropriate neighboring cells
  ft(1)=f(x+1,y,z,1)
  ft(2)=f(x,y+1,z,2)
  …
  gt(1,:)=g(x+1,y,z,1,:)
  gt(2,:)=g(x,y+1,z,2,:)
  …
  do i=1,27
  ! compute density, momentum density
    density+=ft(i)
  …
  end do
  do i=13,27
  ! compute magnetic field
    b(1)+=gt(i,1)
  …
  end do

  do i=1,27
  ! compute feq
    feq(x,y,z,i)=
  …
  end do
  do i=13,27
  ! compute geq
    geq(x,y,z,i,1)=
  …
  end do
  end do: endo do: end do
```

Figure 4 Optimized LB MHD code

additional overhead. The performance of the `mpi_sendrecv` implementation was close enough to that of the original implementation to be within the variability we had seen during several runs, approximately 4%. The second experiment concerns the use of global memory. The NEC MPI implementation reserves an area of memory on each node through which all messages are staged. User defined storage may be allocated in this area either by compiler directives, or by allocating memory via `mpi_alloc_mem`. The code was slightly modified to use the latter approach, and timings were obtained for the 256 and 512 processor runs. Again, the performance fell into the range of times we had observed for the initial version. With hindsight, the reason is quite clear. For messages of this size (roughly 1-2 MB, so latency dominates), compared with the cost of an MPI message, moving data within memory

is extremely fast on the ES. That is, each message is sped up by only a small amount. In addition, looking from the single node performance point of view, for the 512 processor case, assuming we save one memory copy on the send and one on the receive, we eliminate only about 8 GW of copying. This is less than 0.5% of the total vector elements processed during all operations.

Turning to our first production grid of 1024^3 we carried out simulations at 1024, 2048, and 4096 processors. Table 1 shows performance data for these and the previous two benchmarking runs described above.

Table 1 Performance Data for LBMHD obtained via ftrace

Proc.	Grid	Decomp px/py/pz	% MPI comm.	Avg. Msg. Size (MB)	Perf./proc. Gflop/s	VOR	AVL
256	512	16/8/2	7.7	2.1	5.43	99.72	254.2
512	512	16/16/2	9.1	1.1	5.19	99.59	253.2
1024	1024	16/16/4	5.1	2.3	5.44	99.62	254.5
2048	1024	32/16/4	8.6	2.1	5.36	99.71	254.5
4096	1024	32/32/4	-	1.1	5.16	99.58	253.3

For each grid size, with increasing concurrency, the performance per processor drops off slightly. This is mainly the effect of communication overhead increasing, due to both the cost of communication and the increasing ratio of communication to computation. Both of these effects can be seen in the column listing the percentage of time spent in MPI communication. The VOR and AVL values show that the performance of the computational kernel is hardly affected by the scaling up of the problem. The previously discussed experiments were performed with all I/O turned off. Our application can use either MPI I/O or I/O to separate files to record snapshots for visualization, or for saving the final state. Even though the ES can support MPI I/O, because each node has a separate filesystem and MPI I/O is implemented through a software layer on top of this, the separate file I/O strategy proved the most efficient. Saving the full magnetic field and velocity data at 4096 processors took less than an additional 1% wall clock time for a simulation of 5000 iterations (1.5 seconds out of 315). Finally, for one special run using a 1440^3 grid and 4800 processors we ran the simulation at length to probe the onset evolution of turbulence at high resolution. This calculation ran for almost 2 hours at an average performance of 5.47 Gflop/s per processor giving a total aggregate performance of 26.25 Tflop/s. The `ftrace` output is shown is Figure 5.

```
Global Data of 4800 processes:
==============================
                                      Min [U,R]             Max [U,R]         Average
Real   Time (sec)           :     7009.577 [0,1727]     7014.524 [0,3936]     7012.237
User   Time (sec)           :     6980.809 [0,47]       7008.636 [0,58]       7002.136
System Time (sec)           :        0.108 [0,3174]       14.399 [0,2420]        2.096
Vector Time (sec)           :     6750.045 [0,599]      6806.688 [0,1]        6779.005
Instruction Count           : 442363454165 [0,47]   444864848376 [0,6]    444098440561
Vector Instruction Count    : 275326393521 [0,47]   275730111623 [0,1]    275611247439
Vector Element Count        : 66001820265319 [0,900]  66135946388005 [0,4494] 66090298297808
FLOP Count                  : 38290765605961 [0,2404] 38290765620970 [0,0]    38290765607203
MOPS                        :     9448.137 [0,539]      9487.749 [0,1253]     9462.661
MFLOPS                      :     5463.369 [0,58]       5485.147 [0,47]       5468.446
Average Vector Length       :      239.640 [0,299]       239.873 [0,1734]      239.795
Vector Operation Ratio (%)  :       99.745 [0,6]          99.748 [0,242]        99.746
Memory size used (MB)       :     1391.518 [0,2404]     1395.830 [0,0]        1391.831


Overall Data:
=============
Real   Time (sec)           :     7014.524
User   Time (sec)           : 33610251.045
System Time (sec)           :    10060.747
Vector Time (sec)           : 32539225.401
GOPS   (rel. to User Time)  :    45420.738
GFLOPS (rel. to User Time)  :    26248.517
Memory size used (GB)       :     6524.206
```

Figure 5 ftrace output from a 4800 processor 1440^3 grid point simulation

5. Simulation Results

2D LB simulations with scalar distributions for both Navier-Stokes [14] and MHD [15] have been performed and the results compare very well with the corresponding results from conventional CFD spectral codes. The Taylor-Green vortex flow has been extensively studied [16] and is of continual interest computationally because its flow pattern is simple:

$$u_x(\mathbf{x},0) = U_0 \sin(kx)\cos(ky)\cos(kz),$$

$$u_y(\mathbf{x},0) = 0,$$

$$u_z(\mathbf{x},0) = -U_0 \cos(kx)\cos(ky)\sin(kz)$$

yet it undergoes vortex stretching (in the y-drection) and exhibits turbulent decay mechanisms which produce small eddies. The interaction of the Taylor-Green vortex with a magnetic field has been examined in the context of dynamo theory [17]. Here, we choose a somewhat novel initial condition: what is the effect of a Taylor-Green initial magnetic field profile (with no magnetic field component in the y-direction) constraining a criss-cross pattern of Kelvin-Helmholtz unstable vorticity layers at Reynolds numbers in the range of 100 [Reynolds number $\mathrm{Re} = U_0 L / \mu$, and magnetic Reynolds number $\mathrm{Re}_m = U_0 L / \eta$]. The vorticity layers are in the xy-plane and the vorticity tubes are initially uniform in z. After 10 K iterations, the 3D turbulence induces vortex stretching in the z-direction as seen in the view of the vorticity isosurfaces for magnitudes $\geq 0.4\ |\boldsymbol{\omega}|_{\max}$ shown in upper Figure 5 below. Cut-away planes after 10 K iterations of vorticity, corresponding to the isosurface, are shown in lower Figure 5. At 40 K iterations, the vorticity isosurfaces exhibit interesting structures throughout the volume as the vorticity tubes deform, (with the cutaway xy-planes also displayed) as shown in Figure 6. At 70 K iterations, one sees some of the vortex tubes contort further as they tend to radiate outwards in the xy-plane, as seen in the vorticity isosurfaces and corresponding cutaway xy-planes. This is shown in Figure 7.

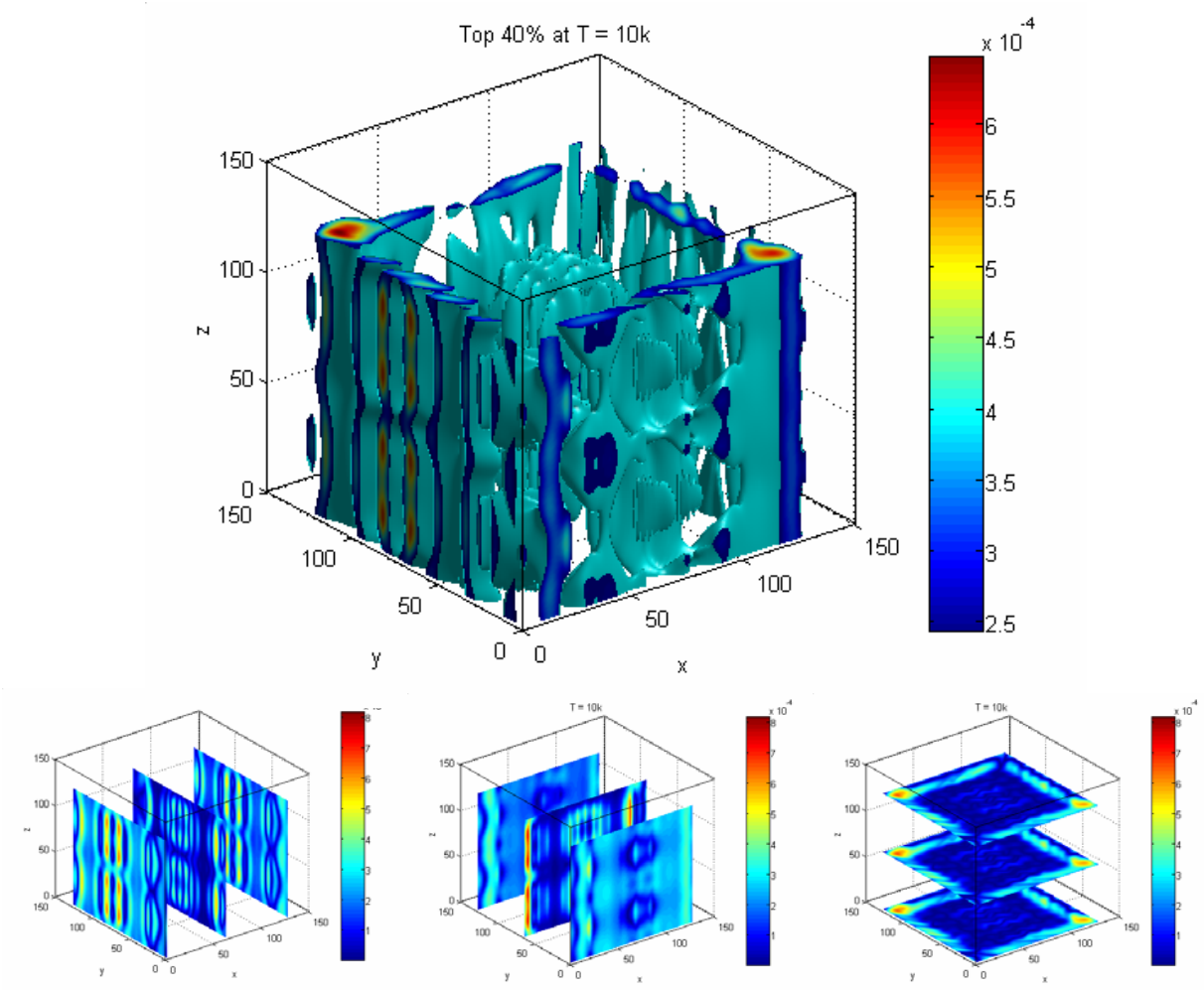
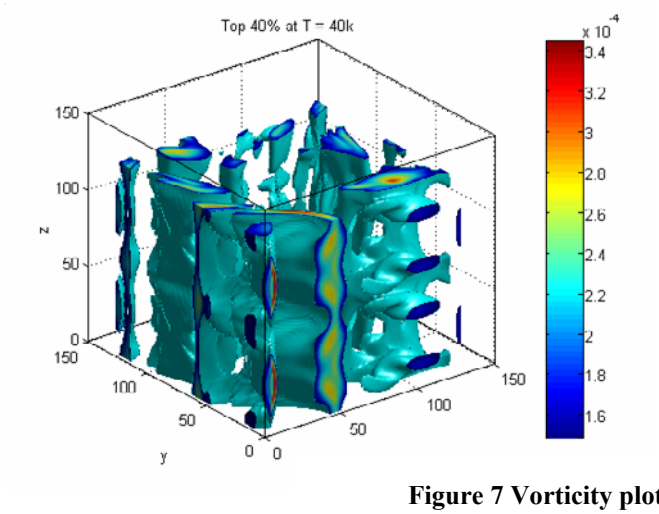


Figure 6 Vorticity plots after 10K iterations



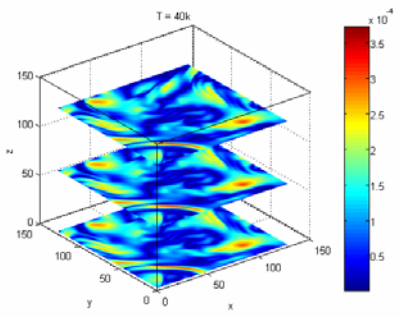
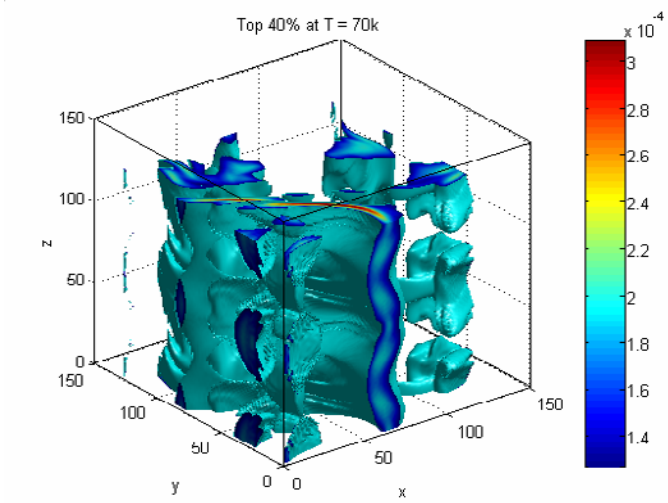


Figure 7 Vorticity plots after 30K iterations



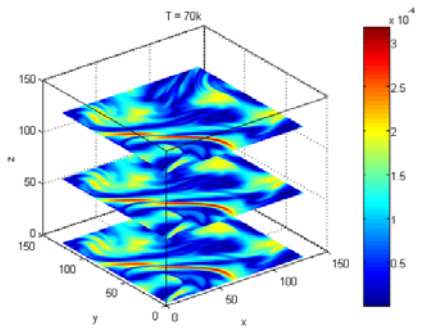


Figure 8 Vorticity plots after 70K iterations

6. Summary

We have presented data on the performance and initial analysis of LB MHD simulations carried out at unprecedented scale and resolution. The preliminary results are very interesting and more time is needed to analyze the results and novel turbulence features exposed in our simulation. In addition, simulations need to be performed at even higher Reynolds and magnetic Reynolds numbers, with even greater resolution.

The simple LBMHD algorithm will now need to be extended to incorporate constraints that enforce the positive-definiteness of the distribution functions. In the Navier-Stokes case, such an entropic algorithm requires a slight augmentation to the collision operator

$$\frac{\partial f(\mathbf{x},\boldsymbol{\xi},t)}{\partial t} + \boldsymbol{\xi}\cdot\nabla f(\mathbf{x},\boldsymbol{\xi},t) = -\alpha(\mathbf{x},t)\,\beta\left[f(\mathbf{x},\boldsymbol{\xi},t) - f^{eq}(\mathbf{x},\boldsymbol{\xi},t)\right]$$

where β is a fixed (tunable) parameter while $\alpha(\mathbf{x},t)$ must be determined by a Newton-Raphson iterative scheme such that the discrete H-function for the system satisfies

$$H[\mathbf{f}] = H\left[\mathbf{f} - \alpha\left(\mathbf{f} - \mathbf{f}^{eq}\right)\right]$$

at each grid point at each time step. It can be shown that $\alpha = 2$ in equilibrium. Our Navier-Stokes entropic simulations have shown that typically one requires less than 5 iterations to obtain convergence with errors of

order 10^{-10}. Since analytic expressions exist for both the H-function and $\mathbf{f}^{eq}$, the Newton-Raphson iterations require only local node information and should be easily vectorized. By extending this entropic algorithm to MHD, we would have an unconditionally stable tool to examine MHD phenomena at arbitrary viscosity and resistivity in arbitrary geometry (since boundary conditions are readily handled by bounce-back rules [2]), ideally suited for vectorization and parallelization. These areas of research are currently under investigation.

7. Acknowledgements

The authors would like to thank S. Kitawaki for his help and assistance during our visit, and Dr. T. Sato of the Earth Simulator Center for granting us access to the ES.

JC and LO were supported by the Director, Office of Science, Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No.DE-AC02-05CH11231. GV and LV were supported by respective grants from DoE.

8. References

[1] D. Biskamp, *Magnetohydrodynamic Turbulence*, Cambridge Univ. Press, 2003

[2] S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*, Clarendon Press, Oxford 2001

[3] P. J. Dellar, J. Comput. Phys. **179**, 95 (2002)

[4] L. Oliker, J. Carter, J. Shalf, D. Skinner, S. Ethier, R. Biswas, J. Djomehri, and R. Van der Wijngaart, Concurrency and Computation Journal: Practice and Experience, **17**, 69 (2005)

[5] A. Macnab, G. Vahala, L. Vahala and P. Pavlo, Proc. 29th EPS Conf. P-1.111 (2002)

[6] P.L. Bhatnagar, E.P. Gross, and M. Krook, Phys. Rev. **94**, 511 (1954)

[7] S. Ansumali, I. V. Karlin and H. C. Ottinger, Europhys. Lett. **63**, 798 (2003).

[8] G. Toth, J. Comput. Phys. **161**, 605 (2000)

[9] Top 500 List: http://www.top500.org

[10] H. Uehara, M. Tamura, and M. Yokohawa, NEC Res. & Develop., **44**, 75 (2003)

[11] G. Wellein, T. Zeiser, S. Donath and G. Hager, Computers and Fluids, Accepted for publication

[12] B. Palmer and J. Nieplocha, Proc. PDCS 2002, 192 (2002)

[13] B.H. Elton, SIAM J. Sci. Comp., 17(4), July 1996

[14] D. Martinez, W. Matthaeus, S. Chen and D. Montgomery, Phys. of Fluids **6**, 1285 (1994)

[15] D. Martinez, S. Chen, W. Matthaeus, Phys. of Plasmas, **1**, 1850 (1994)

[16] M. Brachet, D. I. Meiron, S. A. Orszag, B. G. Nickel, R. Morf, and U. Frisch, J. Fluid Mech. **130**, 411 (1983)

[17] A. Alexakis, P. D. Mininni and A. Poquet, Phys. Rev. E, Accepted (2005)