Measurement of AMR Ideal MHD Parallel Performance

P. Colella
D. F. Martin
N. D. Keen

Applied Numerical Algorithms Group
NERSC Division
Lawrence Berkeley National Laboratory
Berkeley, CA

October 13, 2005

The target platform for this benchmark measurement is a machine named Halem located at GSFC. Halem is the NCCS Compaq AlphaServer SC45 System which consists of 104 symmetric multiprocessor nodes (4 processors per node). Memory is shared within a node.

The Fortran compiler used for this was the native Fortran compiler `f77` with the `-fast` optimization flag. The C++ compiler used was the GNU g++ compiler (version 3.3.1) with flags `-O2 -ftemplate-depth-27`.

The benchmark problem for the ideal MHD code is a simple explosion, with initial conditions $\mathbf{U}(\vec{x},t) = (\rho, \rho u, \rho v, \rho w, B_x, B_y, B_z, e)$:

$$U(\vec{x},0) = \begin{cases} (\rho_1, 0, 0, 0, 0, B_y, 0, e_1) & r < r_0 \\ (\rho_0, 0, 0, 0, 0, B_y, 0, e_0) & r \geq r_0, \end{cases} \tag{1}$$

where r is the distance from $\vec{x}_0$, the center of the spherical explosion, the energy is given by $e = \frac{p}{(\gamma-1)} + \frac{\rho}{2}(u^2+v^2+w^2) + \frac{1}{2}(B_x^2+B_y^2+B_z^2)$, and

$$\begin{aligned} \vec{x}_0 &= (0.4, 0.5, 0.6) \\ r_0 &= 0.1 \\ \rho_0 &= 1 \\ p_0 &= 1 \\ \rho_1 &= 100 \\ p_1 &= 10 \\ B_y &= 10 \\ \gamma &= 1.667 \end{aligned} \tag{2}$$

A sample input used for the runs (for the $64 \times 64 \times 64$ case) is presented in Figure 1.

Table 1 shows the two sizes of benchmark problems used including the respective tagging factor for the undivided gradient of the density, while Table 2 shows the total number of points updated for each run. In all of the benchmark runs, 20 coarse-level timesteps are completed.

Problem size	Density Tagging Factor
64x64x64	0.425
128x128x128	0.2125

Table 1: Baseline Problem Data

The parallel performance of the AMR Ideal MHD code is summarized in Table 3. As we double the linear size of the problem, the computational size of the problem increases by a factor of 4 in 3-dimensions. The factor of 4 increase (rather than a factor of 8 as one

```
godunov.problem = explosion

# Coarsest grid
godunov.num_cells = 64 64 64

# Number of steps, final time, and time step
godunov.max_step = 20
godunov.max_time = 100.0

# initial conditions
godunov.initial_center = 0.4 0.5 0.6
godunov.initial_size = 0.1
godunov.initial_velocity = 0.0 0.0 0.0
godunov.pressure_jump = 100.0
godunov.density_jump = 10.0
# p0, rho0 are ambient pressure, density
godunov.p0 = 1
godunov.rho0 = 1
# specify direction, magnitude of magnetic field
godunov.B_direction = 1
godunov.B_magnitude = 10

#gas properties
godunov.gamma = 1.667
godunov.rgas = 1.0
godunov.wmol = 1.0

# Turn on some output
godunov.verbosity = 2

# Size of the domain's longest dimension
godunov.domain_length = 1.0

godunov.is_periodic = 1 1 1

# Grid refinement
godunov.max_level = 2
# For 3D
godunov.ref_ratio = 2 2 2 2 2

# Regridding parameters
godunov.regrid_interval = 2 2 2 2 2 2
godunov.tag_buffer_size = 3
godunov.refine_thresh = 0.425
```



```
# Grid generation parameters
godunov.block_factor = 4
godunov.max_grid_size = 32
godunov.fill_ratio = 0.75
```

Level	64x64x64	128x128x128
0	5242880	41943040
1	4515968	15318848
2	30875648	110459520
totals	40634496	167721408

Table 2: Number of Points Updated Per AMR Level for each Problem Size

might expect in 3d) is a result of the geometry of this problem. Since the refinement is clustered around the explosion front, it is really the surface area of the front which drives the problem size rather than a volume. (This does, however, highlight another advantage from using local refinement for this problem). So, we can compute scaled efficiency by comparing the run time between two runs which differ by a factor of 2 in base grid size, and a factor of 4 in number of processors. These are shown in Table 4. As can be seen, the scaled efficiencies computed range from 0.71 (71%) to 0.81.

Prob size	Num Procs	Avg Memory MB	Min-Max mem MB	AMR Run secs
64x64x64	4	420	408-430	3268
64x64x64	8	332	273-367	742
64x64x64	16	202	139-251	510
64x64x64	32	114	74-199	483
128x128x128	32	359	321-391	913
128x128x128	64	224	196-261	722
128x128x128	128	149	8.5-228	646

Table 3: Current parallel performance of AMR Ideal MHD code for baseline explosion problem

Base Problem Size	Num Procs	Large Problem Size	Large num processors	Scaled Efficiency
64x64x64	8	128x128x128	32	0.81
	16		64	0.71

Table 4: Scaled Efficiencies computed from Table 3