Parallel Scaling Behaviour of AMR Godunov Application

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

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

January 20, 2005

Target Platform and Compilers

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 work 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 as `-O2 -ftemplate-depth-27`.

For comparison, another platform was used in this work. Seaborg is an IBM SP RS/6000 system located at NERSC/LBL which currently consists of 6,080 processors. Each processor is a POWER3 chip with a clock speed of 375 MHz and peak performance of 1.5 Gflops. The Seaborg processors are clustered into 380 symmetric multiprocessor nodes (16 processors per node). Seaborg is a hybrid system in the sense that memory is distributed among nodes, but within a node memory is shared.

The Fortran compiler used for this work was the AIX Fortran compiler `xlf` version 8.1.1.5 with the flags set to be `-O3 -qarch=auto -qmaxmem=99999 -qhot -qstrict`. The C++ compiler used was the AIX C++ compiler `xlC` version 6.0.0.7 with flags as `-O3 -qstrict -qarch=auto -qstaticinline`.

Need to describe the explosion problem right here. The input used for the runs is presented in Figure 1.

There were four different sizes of the problem used as shown in Table 1.

Problem Size	Refinement Threshold	Problem Name
16x16x16	0.1000	small
32x32x32	0.0500	large
64x64x64	0.0250	ex-large
128x128x128	0.0125	huge

Table 1: Different Sizes of Problems Used in Benchmark

The smallest problem had a base size of 16 cells in all three dimesnions. By increasing the number of cells in each direction by a factor of two, the total work load is nominally increased by a factor of 8. The refinement threshold must also be decreased by a factor of two.

The number of points updated can be used as an approximate measure of the amount of work performed in the calculation. Table 2 shows the number of points updated for each AMR level and for each problem size that was run.

Halem has two different processor speeds. There are batch queues for the slower 1000 MHz processors and the faster 1250 MHz processors. The batch queue for the 1000 MHz proceesors only allowed up to 224 processors, therefore the maximum number of slower processors used together in this work was 128. The maximum number of faster processors

```
godunov.problem = explosion

godunov.gamma = 1.4
godunov.shock_mach = 10.0
godunov.center = 0.6 0.6 0.6
godunov.size = 0.25
godunov.velocity = 0.0 0.0 0.0

godunov.verbosity = 2

godunov.max_step = 4
godunov.max_time = 0.75

godunov.domain_length = 1.0
godunov.num_cells = 64 64 64
godunov.is_periodic = 0 0 0
godunov.max_level = 2
godunov.ref_ratio = 4 4 4

godunov.regrid_interval = 2 2 2
godunov.tag_buffer_size = 3

godunov.refine_thresh = 0.025

godunov.block_factor = 16
godunov.max_grid_size = 16
godunov.fill_ratio = 0.75

godunov.fourth_order_slopes = 1
godunov.flattening = 1
godunov.use_artificial_viscosity = 1
godunov.artificial_viscosity = 0.1

godunov.checkpoint_interval = -1
godunov.plot_interval = -1
godunov.plot_prefix  = plot
godunov.chk_prefix = check

godunov.cfl = 0.8
godunov.initial_cfl = 0.3

godunov.max_dt_growth = 1.1
godunov.dt_tolerance_factor = 1.1
```


Figure 1: Input file for the Unsplit Godunov Explosion Problem

Level	16x16x16	32x32x32	64x64x64	128x128x128
0	16384	131072	1048576	8388608
1	4194304	20316160	73105408	258408448
2	385286144	1303871488	5025136640	18605178880
totals	389496832	1324318720	5099290624	18871975936

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

used together in this work is 512. To run a job with more processors, the halem machine is combined into what is called the "hero mode". In this mode, jobs are executed using some combination of the slower and faster processors.

The performance and memory usage of the code executed on the 1000 MHz processors only is presented in Table 3. The performance and memory usage of the code executed on the 1250 MHz processors only is presented in Table 4. The performance and memory usage of the code executed on a mixture of 1000 MHz and 1250 MHz processors is presented in Table 5.

Table 6 shows parallel performance of the code on Seaborg.

Prob size	Num Procs	Avg Memory MB	Min-Max Memory MB	AMR Run time (sec)
16x16x16	16	265	211-293	398
16x16x16	32	146	118-164	207
16x16x16	64	82	68-93	110
32x32x32	64	243	189-278	360
32x32x32	96	171	138-200	246
32x32x32	128	136	104-159	190

Table 3: Halem Parallel Performance with 1000 MHz Processors

Figure 2 graphically represents key data points from Tables 3, 4, and 5.

The line in the plot represents ideal scaling as problem size and number of processors increase. Data points below the line in a given set indicate better than ideal scaling, which we believe to be as a result of the AMR algorithm efficiency.

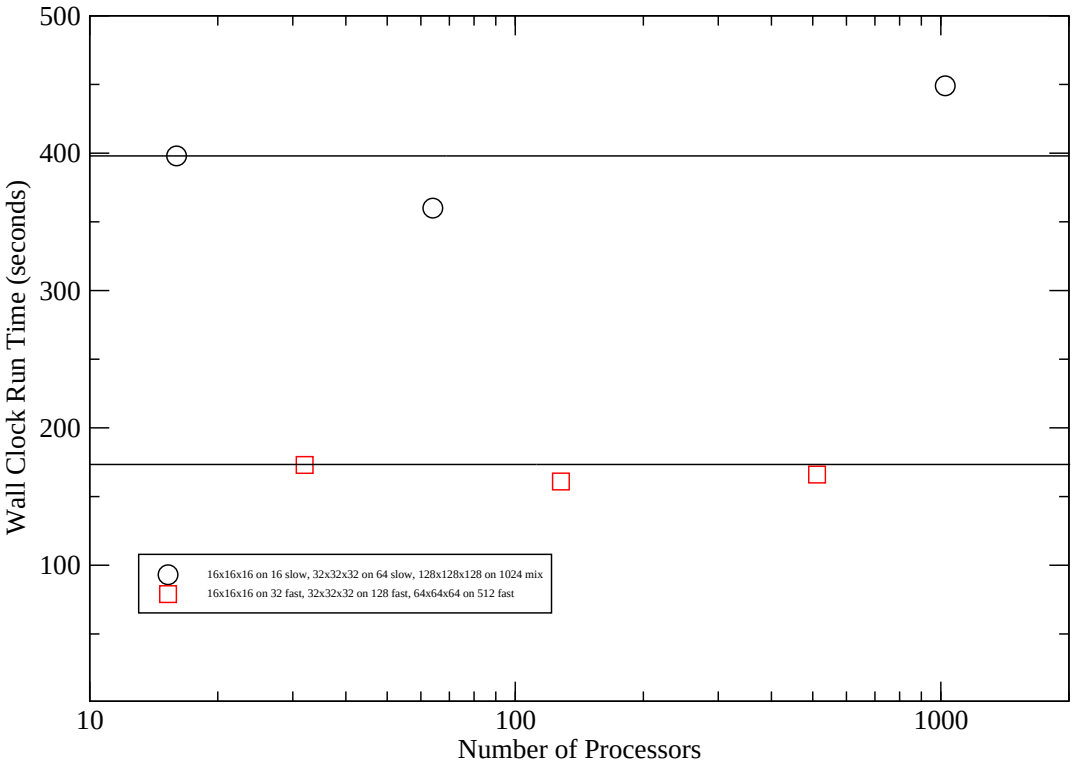

Figure 2: Scaling behavior on Halem as both the problem size and number of processors are increased by a factor of 4 beginning with two different problem sizes. The circles represent 1000 MHz processor results and the hero run. The squares are all results using 1250 MHz processors.

Prob size	Num Procs	Avg Memory MB	Min-Max Memory MB	AMR Run time (sec)
16x16x16	16	264	209-292	325
16x16x16	32	146	117-164	173
32x32x32	64	241	190-276	290
32x32x32	96	171	138-201	209
32x32x32	128	136	104-159	161
64x64x64	128	402	333-459	582
64x64x64	256	227	186-270	318
64x64x64	512	135	111-159	166

Table 4: Halem Parallel Performance with 1250MHz Processors

Prob size	Num Procs	Avg Memory MB	Min-Max Memory MB	AMR Run time (sec)
128x128x128	1024	267	224-332	449
128x128x128	1292	232	196-273	363

Table 5: Halem Parallel Performance with mixed Processors ("Hero" Runs)

Prob size	Num Procs	Avg Memory MB	Min-Max Memory MB	AMR Run time (sec)
32x32x32	32	433	359-493	1558
32x32x32	64	249	195-285	811
32x32x32	128	148	113-171	432
64x64x64	128	411	340-473	1593
64x64x64	256	249	197-291	838
64x64x64	512	168	128-199	449
128x128x128	512	454	392-532	1713
128x128x128	1024	299	250-348	974

Table 6: Seaborg Parallel Performance