AMG

Summary Version

1.0

Purpose of Benchmark

Test parallel performance of unstructured algebraic multigrid methods.

Characteristics of Benchmark

AMG is a parallel algebraic multigrid solver for linear systems arising from problems on unstructured grids. The driver provided for this benchmark builds linear systems for a 3D problem with a 27-point stencil and generates two different problems that are described in section D of the `AMG.readme` file in the `docs` directory. The code is written in ISO standard C.

There are very large demands on main memory bandwidth. Parallelization is accomplished using MPI as well as OpenMP directives. The solve phase is fully threaded, the setup phase, which is much more complicated and contains a large amount of integer arithmetic and branches, is mostly threaded.

See a more detailed description of the code in the `docs/AMG.readme` file.

Mechanics of Building Benchmark

AMG uses a simple Makefile system for building the code. All compiler and link options are set by modifying the `Makefile.include` file appropriately. For large problems, systems with more than 2 billion unknowns, which is the case for the target problems, it is necessary to use the option `-DHYPRE_BIGINT`.

To build the code, after having modified the `Makefile.include` file appropriately, type `make` in the top level `amg` directory.
Other available targets are:

`make clean`	(deletes .o files)
`make veryclean`	(deletes .o files, libraries, and executables)

To configure the code to run with:

1. MPI, add `-DTIMER_USE_MPI` to `INCLUDE_CFLAGS` line in the `Makefile.include` file and use a valid MPI; for additional optimization add `-DHYPRE_CONCURRENT_PERSISTENT`

2. MPI with OpenMP, add `-DTIMER_USE_MPI -DHYPRE_USING_OPENMP` to `INCLUDE_CFLAGS` and add vendor dependent compilation and linking flag for OMP (e.g. `-fopenmp`); for additional optimization add also `-DHYPRE_HOPSCOTCH`

To be able to solve problems that are larger than 2^{31}-1, add `-DHYPRE_BIGINT`

Mechanics of Running Benchmark

All runs described below are to be done using two problems, which are also described in `docs/amg.readme`. The overall problem size is determined by the command line parameters `-n nx ny nz -P Px Py Pz ,` where `nx, ny, nz` define the size per MPI process and `Px, Py, Pz` the process configuration. The two problems are using:

- `-problem 1` (default) will use conjugate gradient preconditioned with AMG to solve a linear system with a 3D 27-point stencil of size `nx*ny*nz*Px*Py*Pz.`

- `-problem 2` simulates a time-dependent problem of size `nx*ny*nz*Px*Py*Pz` with AMG-GMRES. The linear system is also a 3D 27-point stencil. The system is sized to be 5-10% of the large problem.

Example command line parameters to vary the size of the problem (using SLURM notation. –n indicates number of MPI tasks, -N is the number of nodes). To define the number of OpenMP threads per MPI task use `setenv OMP_NUM_THREADS` .

1. Smaller problems:

```
srun -N 32 -n 512 amg -problem 1 \
     -n 96 96 96 -P 8 8 8
srun -N 32 -n 512 amg -problem 2 \
     -n 40 40 40 -P 8 8 8
srun -N 64 -n 1024 amg -problem 1 \
     -n 96 96 96 -P 16 8 8
srun -N 64 -n 1024 amg -problem 2 \
     -n 40 40 40 -P 16 8 8
```

2. Medium-sized problems:

```
srun -N 512 -n 8192 amg -problem 1 \
     -n 96 96 96 -P 32 16 16
srun -N 512 -n 8192 amg -problem 2 \
     -n 40 40 40 -P 32 16 16
srun -N 1024 -n 16384 amg -problem 1 \
     -n 96 96 96 -P 32 32 16
srun -N 1024 -n 16384 amg -problem 2 \
     -n 40 40 40 -P 32 32 16
```

3. Problem (used for CORAL-2 baseline Figure of Merit calculation on BlueGene/Q), Problem 1(default):
`srun -N 24576 -n 393216 amg -n 96 96 96 -P 96 64 64`

4. CORAL-2 class problem:
To create problems that are 4x the size as the problem above, choose `nx, ny, nz, Px, Py, Pz,` so that `nx*Px = 96*96 = 9216 and ny*Py=nz*Pz = 96*128 = 12,288,`
`-n 96 96 96 -P 96 128 128` creates a problem four times as large by using four times as many MPI processes, or
`-n 96 192 192 -P 96 64 64` creates a problem four times as large by quadrupling the size per process.

Figure of Merit (FOM):

There are 3 FOMs printed out at the end of each run of Problem 1:

1. FOM_Setup: nnz_AP / Setup Phase Time (Tier-2)
2. FOM_Solve: nnz_AP * Iterations / Solve Phase Time (Tier-1)
3. Figure of Merit (FOM_1)

`nnz_AP' denotes the sum of the number of nonzeros of the system matrices and interpolation operators on all levels in the algebraic multigrid preconditioner.
`Setup Phase Time' is the wall clock time for the generation of the AMG preconditioner `Solve Phase Time' denotes the wall clock time for solving the problem using AMG-CG after AMG has been set up. `Iterations' denotes the number of iterations to achieve a solution that satisfies the stopping criterion.

FOM_1 is generated as (FOM_Setup + 3*FOM_Solve)/4 .

Note that FOM_Solve is the FOM that needs to be reported.

For Problem 2, which is a Tier-2 problem, the following FOM is printed out at the end of the run:

Figure of Merit (FOM_2)

FOM_2 is generated by nnz_AP * (Iterations + time_steps) / Total Time.

Here 'time_steps' denotes the number of time steps, which is chosen to be 6.
'Iterations' is the cumulative number of iterations across all time steps. `Total Time' is the wall clock time for the whole run, including AMG setup in each time step, AMG-GMRES solve times for all system solves (5 in each time step) and minor matrix manipulations; it does not include the time to set up the original linear system.

Only Tier-1 FOMS, i.e. the Solve FOM for Problem 1, need to be reported. Note that it is permitted to use command line option -keepT, which will store transpose matrices and use matvec instead of a transposed matvec.

Benchmark Verification:

The benchmark delivers correct results if the 'Final Relative Residual Norm' printed out at the end of each run is smaller than 1.e-08 for Problem 1 and smaller than 1.e-10 for Problem 2.

Figure-of-Merit Data on BG/Q

Below, FOM data obtained on Vulcan using 4 OpenMP threads per MPI task are reported, using the following commands:

```
setenv OMP_NUM_THREADS 4
srun -N <nodes> -n <cores> amg -problem 1 -n 96 96 96 \
 -P <px> <py> <px>
srun -N <nodes> -n <cores> amg -problem 2 -n 40 40 40 \
 -P <px> <Py> <pz>
```

where `px, py, pz` are chosen as given for the smaller and medium-sized problems in 1., 2., and the CORAL-2 problem in 3.

For all runs mpixlc_r and the following compile options were used:

```
-O2 -DTIMER_USE_MPI -DHYPRE_USING_OPENMP -qsmp=omp \
-qmaxmem=-1 -DHYPRE_HOPSCOTCH \
-DHYPRE_USING_PERSISTENT_COMM -DHYPRE_BIGINT.
```

The data printed in red are for the problem given in 3. above.

N	n	px	py	pz	its	FOM_setup	FOM_solve	FOM_1	FOM_2
32	512	8	8	8	24	8.944E+08	6.268E+09	4.925E+09	1.807E+08
64	1024	16	8	8	24	1.771E+09	1.252E+10	9.832E+09	3.592E+08
128	2048	16	16	8	25	3.492E+09	2.503E+10	1.965E+10	7.006E+08
256	4096	16	16	16	26	6.753E+09	5.001E+10	3.920E+10	1.357E+09
512	8192	32	16	16	27	1.336E+10	9.995E+10	7.830E+10	2.746E+09
1024	16384	32	32	16	28	2.595E+10	1.997E+11	1.562E+11	5.141E+09
2048	32768	32	32	32	30	4.874E+10	3.936E+11	3.074E+11	9.935E+09
4096	65536	64	32	32	30	9.786E+10	7.906E+11	6.174E+11	2.035E+10
24576	**393216**	**96**	**64**	**64**	**36**	5.113e+11	**4.634e+12**	3.603e+12	1.137e+11