AMG2013

Summary Version

2.3

Purpose of Benchmark

Test single CPU performance and parallel scaling efficiency.

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 various 3D problems that are described in section D of the `AMG2013.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, however this is not the case for the setup phase, which is much more complicated and contains a large amount of integer arithmetic and branches. The primary FOM for CORAL includes only the solve phase.

There are two methods for creating the input matrix, its partitions, and communication arrays that are used to drive the calculation. The first uses a global partitioning with memory requirements that scale with the number of processors. The second does not have this limitation and scales much better for large sizes, where large means >~1K processors. Only the second method should be used for large scale runs (`-DHYPRE_NO_GLOBAL_PARTITION`).

See a more detailed description of the code in the `docs/AMG2013.readme` file.

AMG requires a minimum of 1GB of main memory per MPI task.

Mechanics of Building Benchmark

AMG2013 uses a simple Makefile system for building the code. All compiler and link options are set by modifying the `Makefile.include` file appropriately. It is recommended to use the options
`-DHYPRE_NO_GLOBAL_PARTITION -DHYPRE_LONG_LONG`.

To build the code, after having modified the `Makefile.include` file appropriately, type `make` in the top level `amg2013` 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.
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`)
3. To use the assumed partition (recommended for more one thousand processes or more), add `-DHYPRE_NO_GLOBAL_PARTITION`
4. To be able to solve problems that are larger than 2^{31}-1, add `-DHYPRE_LONG_LONG`

Mechanics of Running Benchmark

All runs described below are to be done using the default problem (Problem 1 as described in `docs/amg2013.readme`). The overall problem size is determined by the command line parameters `-r rx ry rz -P Px Py Pz ,` where `rx, ry, rz` define the size per MPI process and `Px, Py, Pz` the process configuration. There are two distribution types for this problem:

- `-pooldist 0` will lead to each MPI task having a disjoint piece of each of the 8 subdomains of the grid, leading to a larger amount of communication and should only be used for tests on 1 node. Note that the total problem size is 8 times as large as when using the same parameters with `-pooldist 1`: `rx*ry*rz*Px*Py*Pz.`

- `-pooldist 1` should be used for all larger runs, since it will generate a problem partitioning more common in application codes. It requires the total number of MPI tasks to be a multiple of 8. The total problem size here is `rx*ry*rz*Px*Py*Pz/8.`

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. Small problem: single node and/or single CPU
```
srun -N 1 -n 1 amg2013 -pooldist 0 -r 6 6 6
srun -N 1 -n 8 amg2013 -pooldist 1 -r 12 12 12
```

2. Medium problem: (<1K node) job
   ```
   srun -N 512 -n 2048 amg2013 -pooldist 1 \
        -r 12 12 12 -P 16 8 8
   ```

3. Large problem (used for CORAL baseline Figure of Merit calculation on BlueGene/Q):
   ```
   srun -N 4096 -n 65536 amg2013 -pooldist 1 \
        -r 12 12 12 -P 32 16 16
   ```

4. CORAL class problem:
   To create a problem that is 2x the size as the Large problem above, choose `rx, ry, rz, Px, Py, Py, Pz,` so that their product is 2x as large as the product of the values chosen above, e. g.
   `-r 12 12 12 -P 32 32 16` creates a problem twice as large by using twice as many MPI processes,
   `-r 12 24 12 -P 32 16 16` creates a problem twice as large by doubling the size per process.

For an optional larger problem, which is 8x as large, replace `-r 12 12 12` with `-r 24 24 24` in (3) above (and `-r 6 6 6` by `-r 12 12 12` in (1) above).

Figure of Merit (FOM):

There are 2 FOMs printed out at the end of each run:

1. system_size / setup_time
2. system_size * #iterations / solve time

For CORAL, it is sufficient to focus on (2) which is a measure of the performance of one solve cycle and ignores the setup phase.

Note that the parameter 'system_size' used to compute the FOM is not the actual system size of the problem and *does not affect the actual AMG run*. For consistency across all runs it is important that it is used as set in the benchmark (512) and not changed. The actual size used will be a multiple of 384, and is dependent on the refinement factor and number of processes.

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-06.

Additional Figure-of-Merit Data

The following is an additional FOM data collected for a larger version of the AMG2013 problem (**10,616,832 grid points per node**). This is not the problem being used for evaluation, but is provided as an additional data point for vendors who want to provide benchmark results above-and-beyond those requested.

4096	65538	4	**4.286E+10**
4096	32768	8	4.076E+10
4096	16384	16	3.732E+10
4096	8162	32	3.286E+10
4096	4096	64	2.724E+10
512	8192	4	5.419E+09
512	4096	8	5.131E+09
512	2048	16	4.756E+09
512	1024	32	4.191E+09
512	512	64	3.433E+09