AMG2023

Code Description

A. General description:

The AMG2023 benchmark consists of a driver (amg.c), a simple Makefile, and documentation, which are available at https://github.com/LLNL/AMG202 . It requires an installation of hypre 2.27.0 and uses hypre's parallel algebraic multigrid solver BoomerAMG in combination with a Krylov solver to solve two linear systems arising from diffusion problems on a cuboid discretized by finite differences. The problems are set up through hypre's linear-algebraic IJ interface. The problem sizes can be controlled from the command line.

For details on the algorithm and its parallel implementation/performance: see the following papers, which are available at https://github.com/hypre-space/hypre/wiki/Publications :

Van Emden Henson and Ulrike Meier Yang, "BoomerAMG: A Parallel Algebraic Multigrid Solver and Preconditioner", Appl. Num. Math. 41 (2002), pp. 155-177.

Hans De Sterck, Ulrike Meier Yang and Jeffrey Heys, "Reducing Complexity in Parallel Algebraic Multigrid Preconditioners", SIAM Journal on Matrix Analysis and Applications 27 (2006), pp. 1019-1039.

Hans De Sterck, Robert D. Falgout, Josh W. Nolting and Ulrike Meier Yang, "Distance-Two Interpolation for Parallel Algebraic Multigrid", Numerical Linear Algebra with Applications 15 (2008), pp. 115-139.

U. M. Yang, "On Long Range Interpolation Operators for Aggressive Coarsening", Numer. Linear Algebra Appl., 17 (2010), pp. 453-472.

A. H. Baker, R. D. Falgout, T. V. Kolev, and U. M. Yang, "Multigrid Smoothers for Ultraparallel Computing", SIAM J. Sci. Comput., 33 (2011), pp. 2864-2887.

J. Park, M. Smelyanskiy, u. M. Yang, D. Mudigere, and P. Dubey, "High-Performance Algebraic Multigrid Solver Optimized for Multi-Core Based Distributed Parallel Systems", Proceedings of the International Conferencefor High Performance Computing, Networking, Storage and Analysis 2015 (SC15).

R. Li, B. Sjogreen, U. Yang, “A New Class of AMG Interpolation Methods Based on Matrix-Matrix Multiplications”, SIAM Journal on Scientific Computing, 43 (2021), pp. S540-S564, https://doi.org/10.1137/20M134931X

R. Falgout, R. Li, B. Sjogreen, L. Wang, U. M. Yang, “Porting hypre to Heterogeneous Computer Architectures: Strategies and Experiences”, Parallel Computing, 108, (2021), a. 102840

The driver provided with AMG builds linear systems for various 3-dimensional problems, which are described in Section D.

B. Coding:

hypre is written in C. The AMG solver in hypre is an SPMD code which uses MPI. Parallelism is achieved by data decomposition. The driver provided with the AMG benchmark achieves this decomposition by simply subdividing the grid into logical P x Q x R (in 3D) chunks of equal size.

C. Parallelism:

AMG is a highly synchronous code. The communications and computations patterns exhibit the surface-tovolume relationship common to many parallel scientific codes. Hence, parallel efficiency is largely determined by the size of the data "chunks" mentioned above, and the speed of communications and computations on the machine. AMG is also memory-access bound, doing only about 1-2 computations per memory access, so memory-access speeds will also have a large impact on performance.

D. Test problems

Problem 1 (default): The default problem is a 3D diffusion problem on a cuboid with a 27-point stencil. It is solved with AMG-GMRES. To determine when the solver has converged, the driver uses the relative-residual stopping criteria

$$\|r_k\|_2 / \|b\|_2 < 10^{-12}$$

Problem 2 (-problem 2): This problem solves a 3D Laplace problem on a cuboid with a 7-point stencil. To determine when the solver has converged, the driver uses the relative-residual stopping criteria

$$\|r_k\|_2 / \|b\|_2 < 10^{-8}$$

E. Building the Code

The AMG benchmark uses a simple Makefile system for building the driver.

It requires an installation of hypre 2.27.0, which can be downloaded from https://github.com/hypre-space/hypre via:

```
git clone -b v2.27.0 https://github.com/hypre-space/hypre.git
```

Information on how to install hypre is available here: https://hypre.readthedocs.io/en/latest/

Depending on the machine and desired programming models, different configurations are needed. hypre configure options can be obtained by typing

```
./configure --help
```

in hypre's src directory.

For a CPU machine, hypre generally can be easily installed by typing

```
./configure
make install
```

in the src directory.

If OpenMP threading within MPI tasks is desired, it should be configured as follows:

```
./configure --with-openmp --enable-hopscotch
```

If hypre should be run on Nvidia GPUs, use:

```
./configure --with-cuda
```

or

```
./configure --with-cuda --with-device-memory-pool
```

to use the memory pool option included with hypre.

If hypre is to be run on AMD GPUs, use:

```
./configure --with-hip --with-gpu-arch=gfx90a
--with-MPI-lib-dirs="${MPICH_DIR}/lib" --with-MPI-libs="mpi"
--with-MPI-include="${MPICH_DIR}/include"
```

If the problem to be run is larger than 2 billion, i.e., Px*Py*Pz*nx*ny*nz is larger than 2 billion, where Px*Py*Pz isthe total number of MPI tasks and nx*ny*nz the local problem sizeper MPI task, hypre needs to be configured with

```
--enable-mixed-int
```

since it requires 64-bit integers for some global variables. By default, hypre uses 32-bit integers.

To build the code, first modify the 'Makefile' file appropriately, then type

```
make
```

Other available targets are

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

F. Optimization and Improvement Challenges

This code is memory-access bound. We believe it would be very difficult to obtain "good" cache reuse with an optimized version of the code.

G. Parallelism and Scalability Expectations

Previous versions of AMG have been run on the following platforms:

BG/Q - up to over 1,000,000 MPI processes
BG/P - up to 125,000 MPI processes
and more

Consider increasing both problem size and number of processors in tandem. On scalable architectures, time-to-solution for AMG will initially increase, then it will level off at a modest numbers of processors, remaining roughly constant for larger numbers of processors. Iteration counts will also increase slightly for small to modest sized problems, then level off at a roughly constant number for larger problem sizes.

For example, we get the following timing results (in seconds) for a system with a 3D 27-point stencil, distributed on a logical P x Q x R processor topology, with fixed local problem size per process given as 96 x 96 x 96:

P x Q x R	procs	setup time	solve time
8x 8x 8	512	14.91	51.05
16x16x 8	2048	15.31	53.35
32x16x16	8192	16.00	57.78
32x32x32	32768	17.55	65.19
64x32x32	65536	17.49	64.93

These results were obtained on BG/Q using MPI and OpenMP with 4 OpenMP threads per MPI task and configuring hypre with --enable-hopscotch --enable-persistent and --enable-bigint.

To measure strong scalability, it is important to change the size per process with the process topology:

The following results were achieved on RZTopaz for a 3D 7-pt Laplace problem on a 300 x 300 x 300 grid.

```
srun -n <P*Q*R> amg -P <P> <Q> <R> -n <nx> <ny> <nz> -problem 2
```

Using MPI only

P x Q x R	nx x ny x nz	setup time	solve time
1 x 1 x 1	300x300x300	43.37	61.85
2 x 1 x 1	150x300x300	31.06	42.09
2 x 2 x 1	150x150x300	15.68	22.74
2 x 2 x 2	150x150x150	8.44	12.59
4 x 2 x 2	75x150x150	5.37	8.39
4 x 4 x 2	75x 75x150	2.70	6.80

Using MPI with 4 OpenMP threads per MPI task

P x Q x R	nx x ny x nz	setup time	solve time
1 x 1 x 1	300x300x300	17.56	20.81
2 x 1 x 1	150x300x300	12.04	14.48
2 x 2 x 1	150x150x300	6.35	8.78
2 x 2 x 2	150x150x150	3.14	6.84
4 x 2 x 2	75x150x150	2.44	6.73

H. Running the Code

The driver for AMG is called `amg'. Type

```
mpirun -np 1 amg -help
```

to get usage information. This prints out the following:

```
Usage: amg [<options>]

 -P <Px> <Py> <Pz> : define MPI process topology The total amount of
                     MPI processes is Px*Py*Pz.

 -n <nx> <ny> <nz> : define size per MPI process for problems on cube

 -problem <p>      : <p> needs to be 1 (default) or 2

 -printstats       : print out detailed info on AMG preconditioner and
                     number of iterations

 -printallstats    : print out detailed info on AMG preconditioner,
                     number of iterations and residual norms for each
                     iteration

 -printsystem      : print out the system
```

All arguments are optional. An important option for the AMG compact application is the `-P' option. It specifies the MPI process topology on which to run.

The `-n' option allows one to specify the local problem size per MPI process, leading to a global problem size of <Px>*<nx> by <Py>*<ny> by <Pz>*<nz>.

H. Timing Issues

If using MPI, the whole code is timed using the MPI timers. If not using MPI, standard system timers are used. Timing results are printed to standard out and are divided into "Setup Phase" times and "Solve Phase" times for both problems.

I. Memory Usage

AMG's memory needs are somewhat complicated to describe. They are very dependent on the type of problem solved and the AMG options used. When turning on the '-printstats' option, memory complexities <mc> are displayed, which are defined by the sum of non-zeroes of all matrices (both system matrices and interpolation matrices on all levels) divided by the number of non-zeroes of the original matrix, i.e., at least about <mc> times as much space is needed. However, this does not include memory needed for communication, vectors, auxiliary computations, etc.

Figures J1 and J2 provide information about memory usage for Problems 1 and 2 on 1 GPU node of Lassen (equivalent to Sierra) using 4 MPI tasks and 1 node of Quartz using 4 MPI tasks with 9 OpenMP threads each for Problem 1 (Figure J1) and Problem 2 (Figure J2) for increasing problem sizes n x n x n per MPI task. Both Memory Peak, i.e., the largest amount of memory used in the run as well as the accumulated memory usage are listed. The black dashed line indicates the GPU memory available on 1 GPU (V-100) on Lassen.

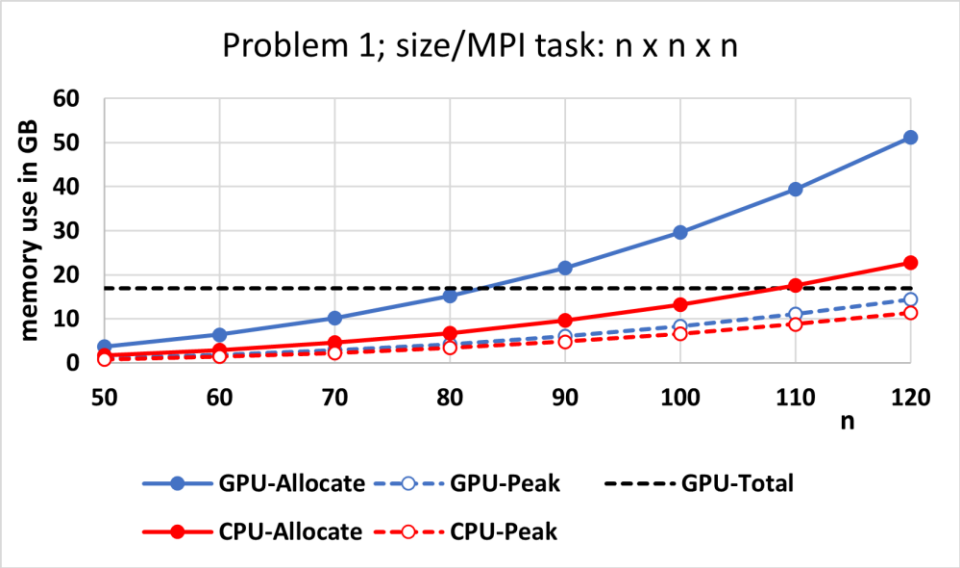

Figure J1. Memory usage for Problem 1 (AMG-GMRES, 27pt stencil) for increasing problem size nxnxn per MPI task.

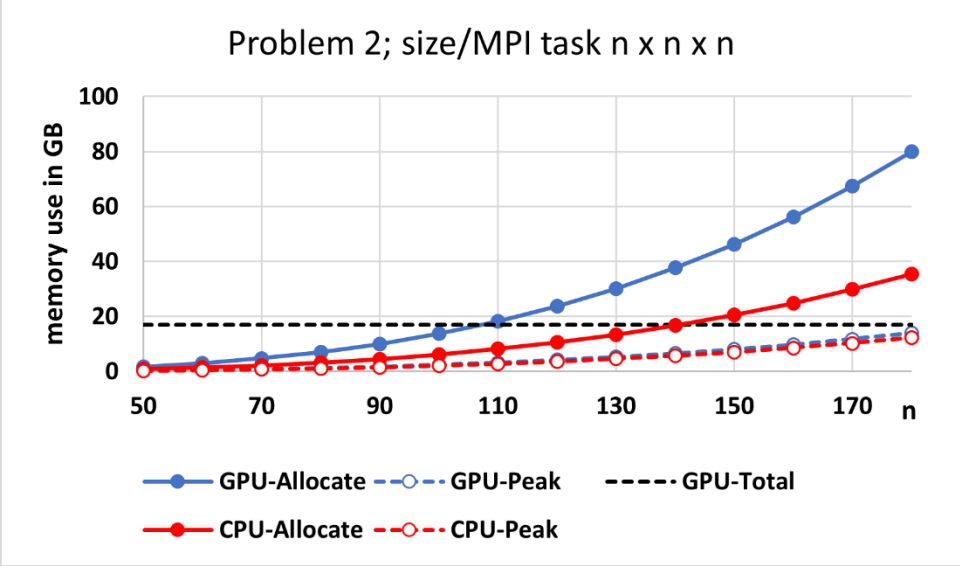

Figure J2. Memory usage for Problem 2 (AMG-PCG, 7pt 3D stencil) for increasing problem size nxnxn per MPI task.

J. Figure of Merit

The figures of merit (FOM_1 and FOM_2) are calculated using the total number of nonzeros for all system matrices and interpolation operators on all levels of AMG, AMG setup wall clock time (FOM_1), and AMG solve phase time and number of iterations (FOM_2).

Currently, FOM is evaluated as follows: FOM = (FOM_1 + FOM_2)/2

Final FOM still TBD.

K. Suggested Test Runs

TBD

L. Expected Results

Consider the following run, which used a hypre installation with MPI and OpenMP on a Linux box for a CPU linking with OpenMP and setting OMP_NUM_THREADS to 3:

```
 mpirun -np 4 amg -P 2 2 1 -n 50 50 50 -printstats
```

This is what AMG prints out:

```
This is what AMG prints out:

Running with these driver parameters:
  solver ID    = 1

  Laplacian_27pt:
    (Nx, Ny, Nz) = (100, 100, 50)
    (Px, Py, Pz) = (2, 2, 1)

=============================================
Generate Matrix:
=============================================
Spatial Operator:
  wall clock time = 0.075466 seconds
  wall MFLOPS     = 0.000000
  cpu clock time  = 0.190000 seconds
  cpu MFLOPS      = 0.000000

  RHS vector has unit components
  Initial guess is 0
=============================================
IJ Vector Setup:
=============================================
RHS and Initial Guess:
  wall clock time = 0.001849 seconds
  wall MFLOPS     = 0.000000
  cpu clock time  = 0.010000 seconds
  cpu MFLOPS      = 0.000000

Solver: AMG-PCG
HYPRE_ParCSRPCGGetPrecond got good precond


 Num MPI tasks = 4  Num OpenMP threads = 3


BoomerAMG SETUP PARAMETERS:
```

```
 Max levels = 25
 Num levels = 5

 Strength Threshold = 0.250000
 Interpolation Truncation Factor = 0.000000
 Maximum Row Sum Threshold for Dependency Weakening = 0.900000

 Coarsening Type = PMIS

 No. of levels of aggressive coarsening: 1

 Interpolation on agg. levels= multipass interpolation
 measures are determined locally


 No global partition option chosen.

 Interpolation = extended+i interpolation

Operator Matrix Information:

            nonzero         entries per row        row sums
lev   rows  entries  sparse  min  max   avg       min         max
=================================================================
 0  500000 13142992  0.000     8   27  26.3   0.000e+00   1.900e+01
 1    7046   174928  0.004     5   42  24.8  -7.567e-13   2.883e+02
 2    1265   106427  0.067    18  142  84.1  -5.451e-13   4.359e+02
 3     170    13374  0.463    31  140  78.7   8.256e+01   6.706e+02
 4      17      287  0.993    16   17  16.9   2.751e+02   1.063e+03



Interpolation Matrix Information:
                 entries/row    min     max         row sums
lev  rows cols    min max     weight   weight     min       max
=================================================================
 0 500000 x 7046    1  10   2.262e-03 9.985e-01 2.119e-01 1.000e+00
 1  7046 x 1265     1   8   4.994e-03 6.801e-01 3.312e-01 1.000e+00
 2  1265 x 170      1   8   2.028e-03 7.890e-01 8.586e-02 1.000e+00
 3   170 x 17       0   8  -4.880e-02 1.668e-01 0.000e+00 1.000e+00


     Complexity:    grid = 1.016996
                 operator = 1.022447
                 memory = 1.090040



BoomerAMG SOLVER PARAMETERS:

  Maximum number of cycles:         1
  Stopping Tolerance:               0.000000e+00
  Cycle type (1 = V, 2 = W, etc.):  1
```

```
  Relaxation Parameters:
   Visiting Grid:                     down   up  coarse
            Number of sweeps:           2    2     1
   Type 0=Jac, 3=hGS, 6=hSGS, 9=GE:    18   18    18
   Point types, partial sweeps (1=C, -1=F):
                  Pre-CG relaxation (down):   0   0
                   Post-CG relaxation (up):   0   0
                             Coarsest grid:   0

=============================================
Problem 1: AMG Setup Time:
=============================================
PCG Setup:
  wall clock time = 0.258020 seconds
  wall MFLOPS     = 0.000000
  cpu clock time  = 0.680000 seconds
  cpu MFLOPS      = 0.000000


nnz_AP / Setup Phase Time: 5.552436e+07

=============================================
Problem 1: AMG-PCG Solve Time:
=============================================
PCG Solve:
  wall clock time = 1.316479 seconds
  wall MFLOPS     = 0.000000
  cpu clock time  = 3.850000 seconds
  cpu MFLOPS      = 0.000000


Iterations = 20
Final Relative Residual Norm = 7.979902e-09


nnz_AP * Iterations / Solve Phase Time: 2.176471e+08



Figure of Merit (FOM_1): 1.632353e+08
```

For further information on AMG contact
Ulrike Yang
email: umyang@llnl.gov