PENNANT

Mini-app that operates on 2-D, unstructured, finite element meshes with arbitrary polygons

Description

PENNANT is a mini-app that operates on 2-D, unstructured, finite element meshes with arbitrary polygons.

PENNANT implements a Lagrangian staggered-grid hydrodynamics algorithm on a 2-D unstructured finite-volume mesh composed of arbitrary polygons with arbitrary connectivity. It makes heavy use of indirect addressing and irregular memory access patterns. It is written in standard C++ and supports MPI and OpenMP parallelism, and a (partially developed) CUDA implementation is also available for reference.

More detailed documentation can be found in the doc directory of the PENNANT distribution.

How to Build PENNANT

A simple Makefile is provided in the top-level directory for building the code. Before using it, you may wish to edit the definitions of CXX and CXXFLAGS to specify your desired C++ compiler and flags, and to choose between optimized/debug and serial/OpenMP/MPI builds. Then a simple "make" command will create a build subdirectory and build the PENNANT binary in that directory.

PENNANT has been tested under GCC, PGI, Intel, and IBM compilers.
Building under other compilers should require at most minor changes.

The ACES benchmark tests were run with the tagged version 0.9 of PENNANT on https://github.com/lanl/PENNANT . (Note that, between that version and the June 2018 head, the source code is unchanged; only tests and documentation have been added.)

How to Run PENNANT

Several test problems are provided in subdirectories under the top-level test directory. Each subdirectory contains a small text file <testname>.pnt containing run parameters for that test case. The command line

pennant <testname>.pnt

is used to run a test in serial mode. If running under MPI, this should be preceded by mpirun or similar command as appropriate on your system. Use the -np argument (or equivalent) to change the number of MPI ranks, and the code will determine a suitable decomposition. Similarly, for OpenMP, specify the OMP_NUM_THREADS environment variable.

Crossroads Benchmark Problems

The Crossroads Benchmarks website defines problem sizes to be used in benchmarking.
The PENNANT test problems corresponding to this definition are:

"Small" problem: leblancbig
"Medium" problem: leblancx4
"Large" problem: leblancx64

In addition, the nohpoly problem may be used for verifying that the baseline code functionality has been preserved (see below). Note for reference that nohpoly would also be considered a "small" problem by the RFP definition.

Correctness

Sample outputs for the Crossroads test problems are given in the sample_outputs directory of the PENNANT distribution. These outputs are from earlier runs on the Edison cluster at NERSC, and can be found in the test/sample_outputs/edison directory. More recent runs on Trinity (Haswell) gave similar outputs, though with different runtimes:

leblancbig	1 node	2.047821e+01
leblancx4	16 nodes	8.302526e+01
leblancx64	4096 nodes	1.529997e+03

Results can be verified by comparing the final "energy check" diagnostics, printed near the end of the standard output, to those in the sample output file. All three quantities listed (total, internal, and kinetic energy) should match to within a relative error of 10^-5.

PENNANT-Specific Run Rules

The purpose of the PENNANT benchmark is to demonstrate performance on an application using general unstructured meshes, that is, meshes which contain arbitrary polygons with arbitrary connectivity between them. Any code changes made by the vendor must preserve this capability. That is, although the leblanc* benchmark problems happen to use meshes with an underlying Cartesian structure, the code may not be modified in any way that takes advantage of this structure. This can be verified by ensuring that any modified version can correctly run the nohpoly test problem, in which the mesh is truly unstructured.

For the optimized version, vendors are allowed to make other modifications to the mesh (new data structures, element numbering, MPI decomposition,...) so long as support for general unstructured meshes is preserved. For both the base and optimized versions, vendors are also allowed to tune the "chunksize" input parameter to a value best suited to their architecture.

The problem size for the Crossroads benchmark was chosen to give a zone/rank count typical of production runs of full multi-material, multi-physics applications. In production, the PENNANT benchmark uses only a small fraction of system memory, with the remaining memory filled using more materials and more physics (not available in PENNANT) rather than more zones.

Vendor submissions must not attempt to fill memory by running a larger problem size, or consuming all of the node memory and running the given problem on a small number of nodes. Instead, submissions must run the leblancx64.pnt reference problem distributed on a set of

nodes providing at least ~400 TB total memory. (For reference, the benchmark run on 4096 Trinity Haswell nodes had 512 TB available.)

Figure of Merit (FOM) Computation

The FOM for PENNANT benchmarks is total zones processed per second, which can be computed from three lines in PENNANT standard output:

- number of zones in mesh: in the line "Zones:" near the top
- number of cycles run: in line "cycle =" near the bottom
- run time: in line "hydro cycle run time=" near the bottom (note that this time excludes initialization and finalization stages)

For the Crossroads leblancx64 benchmark on Trinity (Haswell):
FOM = zones * cycles / time
= 943718400 * 236621 / 1.529997e+03
= 1.459503e+11

Reporting

For the electronic submission, include all the source and the makefiles used to build on the target platform. Include all standard output files.

Authorship

PENNANT was developed at Los Alamos National Laboratory.