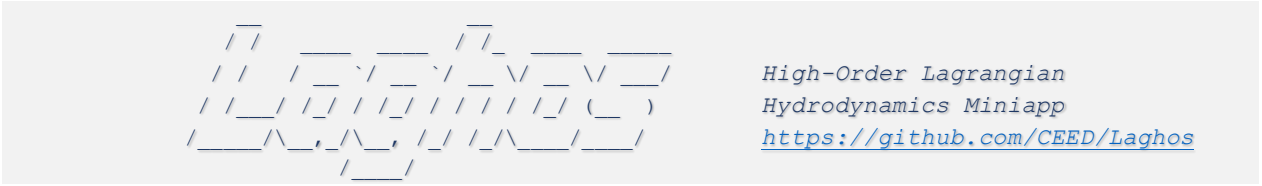


Summary Version

Results and benchmark tarball are for the cts2 version from the CEED/Laghos repository.

Purpose of Benchmark

Laghos (LAGrangian High-Order Solver) is a miniapp that solves the time-dependent Euler equations of compressible gas dynamics in a moving Lagrangian frame using unstructured high-order finite element spatial discretization and explicit high-order time-stepping.

Laghos is based on the discretization method described in the following article:

> V. Dobrev, Tz. Kolev and R. Rieben
> High-order curvilinear finite element methods for Lagrangian hydrodynamics
> *SIAM Journal on Scientific Computing*, (34) 2012, pp. B606-B641

Laghos captures the basic structure of many other compressible shock hydrocodes, including the BLAST code at Lawrence Livermore National Laboratory. The miniapp is built on top of a general discretization library, MFEM, thus separating the pointwise physics from finite element and meshing concerns.

Laghos is a LLNL ASC co-design mini-app that was developed as part of the CEED software suite, a collection of software benchmarks, miniapps, libraries and APIs for efficient exascale discretizations based on high-order finite element and spectral element methods. See http://github.com/ceed for more information and source code availability.

The CEED research is supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, in support of the nation's exascale computing imperative.

Characteristics of Benchmark

In each time step, the problem is ultimately formulated as solving a big system of ordinary differential equations (ODE) for the unknown (high-order) velocity, internal energy and mesh nodes (position). The left-hand side of this ODE is controlled by *mass matrices* (one for velocity and one for energy), while the right-hand side is constructed from a *force matrix*.

```
   __                __
  / /   ____  ____  / /_  ____  _____
 / /   / __ `/ __ `/ __ \/ __ \/ ___/       High-Order Lagrangian
/ /___/ /_/ / /_/ / / / / /_/ (__  )        Hydrodynamics Miniapp
/_____/\__,_/\__, /_/ /_/\____/____/        https://github.com/CEED/Laghos
            /____/
```

Laghos supports two options for deriving and solving the ODE system, namely the *full assembly* and the *partial assembly* methods. **Partial assembly is the main algorithm of interest for this benchmark.**

Other computational motives in Laghos include the following:

- Support for unstructured meshes, in 2D and 3D, with quadrilateral and hexahedral elements (triangular and tetrahedral elements can also be used, but with the less efficient full assembly option). Serial and parallel mesh refinement options can be set via a command-line flag.
- Explicit time-stepping loop with a variety of time integrator options. Laghos supports Runge-Kutta ODE solvers of orders 1, 2, 3, 4 and 6, as well as a specialized Runge-Kutta method of order 2 that ensures exact energy conservation on fully discrete level (RK2Avg).
- Continuous and discontinuous high-order finite element discretization spaces of runtime-specified order.
- Moving (high-order) meshes.
- Separation between the assembly and the quadrature point-based computations.
- Point-wise definition of mesh size, time-step estimate and artificial viscosity coefficient.
- Constant-in-time velocity mass operator that is inverted iteratively on each time step. This is an example of an operator that is prepared once (fully or partially assembled) but is applied many times. The application cost is dominant for this operator.
- Time-dependent force matrix that is prepared every time step (fully or partially assembled) and is applied just twice per “assembly”. Both the preparation and the application costs are important for this operator.
- Domain-decomposed MPI parallelism.
- Optional in-situ visualization with GLVis and data output for visualization / data analysis with VisIt.

Mechanics of Building the Benchmark

For this run, Laghos has the following external dependencies:

- ***hypre***, used for parallel linear algebra, we recommend version 2.10.0b https://computation.llnl.gov/casc/hypre/software.html

- **MFEM**, used for (high-order) finite element discretization, GitHub tag laghos-cts2 https://github.com/mfem/mfem

- **METIS**, used for parallel domain decomposition, we recommend version 4.0.3 http://glaros.dtc.umn.edu/gkhome/metis/metis/download

- **RAJA**, used for handling the kernel's for-loop bodies https://github.com/LLNL/RAJA

Build ***hypre*** from https://computation.llnl.gov/projects/hypre-scalable-linear-solvers-multigrid-methods/download/hypre-2.10.0b.tar.gz:

```
~> tar xzvf hypre-2.10.0b.tar.gz
~> cd hypre-2.10.0b/src/
~/hypre-2.10.0b/src> ./configure --disable-fortran
~/hypre-2.10.0b/src> make -j
~/hypre-2.10.0b/src> cd ../..
```

Build **metis** from http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/OLD/metis-4.0.3.tar.gz:

```
~> tar xzvf metis-4.0.3.tar.gz
~> cd metis-4.0.3
~/metis-4.0.3> make
~/metis-4.0.3> cd ..
~/metis-4.0.3> mv metis-4.0.3 metis-4.0
```

Build **RAJA** from https://github.com/LLNL/RAJA/releases/download/v0.7.0/RAJA-0.7.0.tar.gz:

```
~> tar xzvf RAJA-0.7.0.tar.gz
~> cd RAJA-0.7.0
~/RAJA-0.7.0> mkdir build && cd build
~/RAJA-0.7.0/build> cmake -DENABLE_OPENMP=OFF \
                      -DCMAKE_INSTALL_PREFIX=../../raja ..
~/RAJA-0.7.0/build> make && make install
~/RAJA-0.7.0/build> cd ../..
```

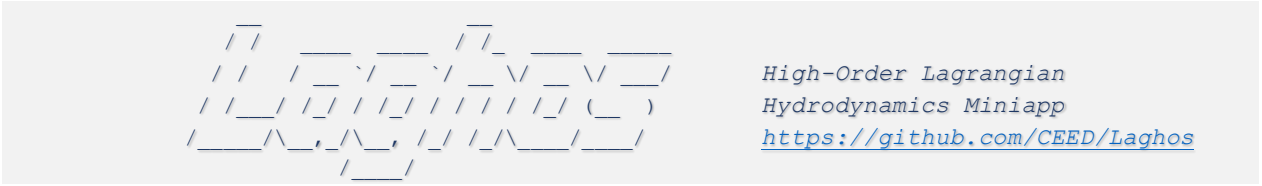
Build **MFEM** from: https://github.com/mfem/mfem/archive/laghos-cts2.tar.gz:

```
~> tar xzvf mfem-laghos-cts2.tar.gz
~> cd mfem-laghos-cts2/
~/mfem-laghos-cts2> make parallel MFEM_USE_RAJA=YES -j
~/mfem-laghos-cts2> cd ..
~/mfem-laghos-cts2> mv mfem-laghos-cts2 mfem
```

Build **Laghos** from: https://github.com/CEED/Laghos/archive/cts2.tar.gz

```
~> tar xzvf Laghos-cts2.tar.gz
~> cd Laghos-cts2/
~> make -j
```

This can be followed by `make test` to check the build respectively.
See `make help` for additional options.



Mechanics of Running the 2D Benchmark

Sedov Blast

The main problem of interest for Laghos is the Sedov blast wave (`-p 1`) with partial assembly option (-pa) and the RAJA backend, accessible through the `okina` options (`-o -q -ra`).

A sample runs in 2D is:

```
mpirun -n 4 laghos -p 1 -m data/square01_quad.mesh -rs 2 -o -q -ra
```

To partition an initial 2D mesh in a way that results in a perfectly balanced partitioning, with each MPI task having the same number of zones, one needs to specify the correct partitioning (`-c 'X Y'`) and initial mesh (`-m`) options. The cartesian partitioning option specifies the relative ratio between the number of MPI tasks in each of the (X, Y) directions.
Furthermore, the number of serial refinements (option `-rs`) should be sufficiently high to produce at least one zone per MPI task, before the parallel refinements (option `-rp`) are performed. The `data/square_10x9_quad` mesh has initially 90 zones and with 36 MPI tasks, one should use at least one serial refinement and the (4,9) cartesian partitioning:

```
mpirun -n 36 laghos -p 1 -m data/square_10x9_quad.mesh -rs 1 -rp 0 -o
-q -ra -c '4 9'
```

Other options allow to run on different ranks with a perfectly balanced partitioning: **64** (`-rs 3 -rp 0 -c '8 8'`), **90** (`-rs 1 -rp 2 -c '10 9'`), **96** (`-rs 2 -rp 1 -c '8 12'`), **120** (`-rs 1 -rp 2 -c '20 6'`) and **128** (`-rs 3 -rp 0 -c '16 8'`).

Verification of Results

To make sure the results are correct, we tabulate reference final iterations (`step`), time steps (`dt`) and energies (`|e|`) for the following runs:

```
mpirun -np 4 laghos -p 1 -m data/square01_quad.mesh -rs 3 -o -q -ra
```

```
mpirun -np 4 laghos -p 1 -m data/square_10x9_quad.mesh -rs 0 -ok 3 -ot
2 -s 7 -o -q -ra
```

run	step	dt	e
1	1064	0.001578	50.3862948290
2	952	0.000240	45.6800237914

An implementation is considered valid if the final energy values are all within round-off distance from the above reference values.

```
   __                __
  / /   ____  ____  / /_  ____  _____
 / /   / __ `/ __ `/ __ \/ __ \/ ___/       High-Order Lagrangian
/ /___/ /_/ / /_/ / / / / /_/ (__  )        Hydrodynamics Miniapp
/_____/\__,_/\__, /_/ /_/\____/____/        https://github.com/CEED/Laghos
            /____/
```

Performance Timing and FOM

Each time step in Laghos contains 3 major distinct computations:

1. The inversion of the global kinematic mass matrix (CG H1).
2. The force operator evaluation from degrees of freedom to quadrature points (Forces).
3. The physics kernel in quadrature points (UpdateQuadData).

By default, Laghos is instrumented to report the total execution times and rates, in terms of millions of degrees of freedom per second (megadofs), for each of these computational phases. These rates are reported as three separate figures of merits in the table below, together with a total combined execution rate which is the reportable **Figure of Merit (FOM)** for the benchmark:

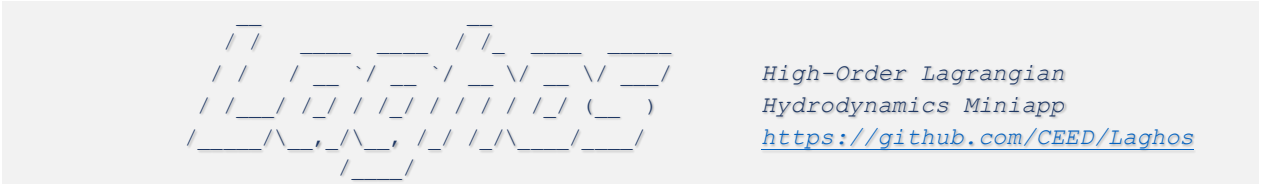
nodes	cores	order	Total dofs/node	FOM_1	FOM_2	FOM_3	FOM
1	36	Q3-Q2	119,468,548	184.054	341.243	170.887	**190.272**

These results were obtained with the following setup on the Quartz machine at LLNL with the `gcc/8.1.0` and `openmpi/4.0.0` modules. The vendor should provide a performance comparison to this run:

```
srun -N1 -n36 ./laghos -p 1 -m data/square_10x9_quad.mesh -rs 2 \
     -rp 5 -pa -ok 3 -ot 2 -s 7 -o -q -ra -ms 4 -c '4 9'
```

Contact

You can reach the Laghos team by emailing laghos@llnl.gov or by leaving a comment in the GitHub issue tracker.



Copyright

The following copyright applies to each file in the CEED software suite, unless otherwise stated in the file:

> Copyright © 2017, Lawrence Livermore National Security, LLC. Produced at the Lawrence Livermore National Laboratory. LLNL-CODE-734707. All Rights reserved.

See files LICENSE and NOTICE for details.

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344, LLNL-TR-770220.