Benchmark test case using OpenSBLI

Satya P Jammy

Aerodynamics and Flight Mechanics Research group, University of Southampton, U.K

The benchmark test case setup using OpenSBLI is the Taylor-Green vortex problem in a cubic domain of length 2 π. The code was setup to compute the validation, strong scaling and weak scaling simulations across architectures. The rest of the document explains setting up the various types of simulations, dependant libraries, compiling and running the simulation.

1 Simulations

OpenSBLI benchmark test case uses an input file (filename is fixed as **input** and its location is the working directory). The input file contains the control parameters for the **type of simulation**, **number of grid points**, **number of iterations**, **number of processors**, and **write output** (HDF5 file). The format of the input file is,

```
type_of_simulation number_of_grid_points_per_direction number_of_iterations
    number_of_processes write_HDF5
```

and the corresponding data types and sizes are,

```
string(size2) int int int string
```

The output written by all the simulations include the following

- The type of simulation, number of grid points per direction, total number of grid points (nx^3)
- The domain decomposition, if applicable i.e. if using MPI
- Total wall-time of the time loop (number of iterations)
- run-time per iteration, i.e. wall-time/number of iterations
- If HDF5 file I/O is specified the time taken for writing the file

1.1 Validation

To validate the code the user needs to specify the type of simulation as **va**, the inputs (2-4) will not have an effect on the simulation. The number of grid points and iterations are predefined as 128 and 500 respectively.

1.1.1 Set-up example

An example input file to run the validation simulation without writing HDF5 output file would be,

```
va 256 10 12 False
```

In the above example, the user can activate HDF5 I/O by setting the last parameter to **True**, in such a case the input file would be,

```
va 256 10 12 True
```

1.1.2 Output

Along with the common outputs specified at the start of the simulation the sum of kinetic energy, enstrophy and density are printed to the screen. If the numbers given by the simulation match the following line then the simulation is considered as valid.

```
0.84625, 261319, 844254, 2.09715e+06
```

1.2 Strong scaling

To perform strong scaling simulations the type of simulation (first input) in the input file should be **ss**. The second and third inputs should be set by the user. The second input corresponds to the number of grid points in each direction (we carried out strong scaling on ARCHER by setting this number as 1024). The simulation will be run for $second\_inp^3$ gridpoints for $third\_input$ iterations. The $fourth\_input$ will have no effect in this simulation and the $fifth\_input$ can be used for I/O tests (if required set this to 0).

1.2.1 Set-up example

An example, input file to do strong scaling simulations using a grid size of 1024^3, for 100 iterations without file I/O would be,

```
ss 1024 100 0 False
```

1.2.2 Output

The output for the strong scaling tests are same as outlined at the starting of the section.

1.3 Weak scaling

Similarly to perform the weak scaling simulations, the first input would be **ws**, the second input will correspond to the number of grid points per process. The third input is the number of iterations for the simulations. The fourth input is the number of processors the user wants the weak scaling simulation to be performed. Depending on this number the dimensions of the problem will be scaled, this is done to avoid very fine grid at higher processor count.

The grid size of the weak scaling simulations would be $(second\_input * fourth\_input)^3$, running on $fourth\_input$ number of processors for $third\_input$ iterations.

1.3.1 Set-up example

An example input file to do the weak scaling simulation with 32^3 grid points per process using 64 MPI processes for 100 iterations would be,

```
ws 32 100 64 False
```

1.3.2 Output

The output for the weak scaling tests are same as outlined at the starting of the section.

2 Compiling

OpenSBLI depends on the OPS framework. OPS should be compiled before compiling the benchmark application. The OPS version that is tested for the current setup is provided along with the source files in the directory *OpenSBLI/OPS/* follow the instructions in the README file in this directory to install OPS. Further details can be found at OPS website , the latest source code of OPS can be downloaded from the github repository link there .

To compile the benchmark test case user should modify the makefile provided in the application directory of OpenSBLI (*OpenSBLI/benchmark/*) by setting the correct compiler flags for the architecture used along with the OPS library install path.

Compiling the benchmark case on MPI and CUDA can be done using the following commands,

```
$make OpenSBLI_mpi
$make OpenSBLI_cuda
```

The current architectures that are supported and tested by the OPS library are the following,

```
MPI
OpenMP
MPI+OpenMP (Hybrid)
CUDA
CUDA+MPI
OpenCL
MPI+OpenCL
OpenACC
MPI+OpenACC
```

Once the application is compiled, it can be run on the architecture.

The benchmark application is translated to various architectures using the OPS translator and is provided and no translation is required for the benchmark test case as specified in the user guide at OPS website. If needed the user can add the C code for any other timer calls directly into *taylor_green_vortex_ops.cpp*.

3 Example output

The example output printed to the screen when running the validation simulation on NVIDIA Tesla K40 using CUDA is given below,

```
$./OpenSBLI_cuda
--------------------------------------------------------------------------------
This is the benchmark application developed for the CFD application of the
UK HPC benchmark suite. The compressible Navier-Stokes equations are solved
on a Cartesian mesh using fourth order central finite difference scheme
and three stage Runge-Kutta time stepping scheme.

     Authors Dr. Satya P Jammy and Prof. Neil Sandham


--------------------------------------------------------------------------------
Some simulation information
The simulation performed is the validation simulation
The number of grid points used per direction are 128
The total number of grid points are 2097152

 Using CUDA device: 0 Tesla K40c

 16/48 L1/shared
End of the simulation information
--------------------------------------------------------------------------------
Iteration is 0
Iteration is 100
Iteration is 200
Iteration is 300
Iteration is 400
End of time iteration loop
performed 500 iterations


--------------------------------------------------------------------------------

The following line should match with the one provided in the document for
validation
0.84625, 261319, 844254, 2.09715e+06

Timings are:
----------------------Simulation time---------------------------------------------
Total Wall time of the time iteration loop for 500 iterations, 33.820737
Time taken for 1 iteration, 0.067641

Not performing file I/O as write HDF5 is set to False
```