EuroHPC JU Benchmark Access

Final Report

Peer-Review Office – V0.1 – 22/11/2018

1 General Information

Type of project granted: EuroHPC JU Benchmark Access.

Tests to obtain the relevant parameters necessary when applying to future EuroHPC JU calls for Regular Access.

1.1 Proposal ID

EHPC-BEN-2022B07-151

1.2 Period of access to the EuroHPC JU facilities

August 2022 – October 2022

1.3 Name of the EuroHPC JU facility assigned

LUMI-C

2 Project information

2.1 Project name to which the tested code corresponds

Scalability of a Large Eddy Simulation Immersed Boundary solver on LUMI-C

2.2 Research field

☐	Biochemistry, Bioinformatics and Life sciences	☐	Fundamental Physics
☒	Chemical Sciences and Materials	☐	Linguistics, Cognition and Culture
☐	Earth System Sciences	☐	Mathematics and Computer Sciences
☐	Economics, Finance and Management	☐	Physiology and Medicine

[✓] Engineering

[ ] Universe Science

[ ] Fundamental Constituents of Matter

2.3 Institutions and research team members

Antonio Posa, CNR-INM, Institute of Marine Engineering, National Research Council of Italy

Riccardo Broglia, CNR-INM, Institute of Marine Engineering, National Research Council of Italy

2.4 Summary of the project interest

The present project is aimed at demonstrating the scalability of an in-house, high-fidelity Large-Eddy Simulation, Immersed-Boundary solver in MPI Fortran language on the LUMI-C cluster, in order to provide evidence of its portability and suitability for future production runs on this system. The results of strong and weak scaling tests will serve for preparing a future proposal to the coming cut-off date of the EuroHPC JU Regular Access (November 2022), with the purpose of studying innovative, bio-inspired propellers, in the framework of an EU-funded research project. These propellers feature tubercles at the leading edge of their blades, as those on the fins of whales. Their purpose is achieving improved performance, compared to conventional design, by disrupting the coherence of the vortices generated at the leading edge of the propeller blades. Finite-differences are utilized to discretize the filtered Navier-Stokes equations. An immersed-boundary methodology enables the use of regular grids, as Cartesian or cylindrical, making the decomposition of the overall flow problem into subdomains very straightforward, efficient and suitable to parallel computing. Communications across subdomains are handled via calls to MPI libraries. I/O operations are performed using calls to parallel HDF5 libraries. The solver is not I/O intensive, with I/O operations taking only about 5% of the overall computational cost of a typical simulation. Although the scalability of the present solver was already tested on several, different architectures, also part of the PRACE and EuroHPC JU infrastructures (Marconi KNL, Joliot-Curie KNL, Joliot-Curie SKL, Joliot-Curie Rome, MareNostrum 4, Vega CPU, Karolina), the test-case that will be considered in this project will be specifically designed to be representative of the computational effort of the problem we aim to tackle in the framework of our next proposal for EuroHPC JU Regular Access.

3 Main features of the code

3.1 Name of the code

Eddy (in house Large-Eddy Simulation Fortran solver)

3.2 Type of the code distribution

In-house academic solver

3.3 Computational problem executed

Filtered Navier-Stokes equations

3.4 Computational method

Finite-differences, fractional-step and immersed-boundaries

3.5 Kind of parallelism used

MPI

3.6 Main libraries used, version and language. Did you use the /usr/local one?

BLAS, LAPACK and HDF5 already available on LUMI-C through modules:

- Cray-libsci/22.08.1.1
- Cray-hdf5-parallel/1.12.1.5

3.7 Which other software did you use on the PRACE machines? Did you use some post-processing or pre-processing tools?

No additional software was required for testing the scalability of the solver.

4 Compilation step

4.1 How is the program compiled?

A makefile was utilized.

4.2 Difficulties met to compile, if any, and how they were tackled.

Compilation was straightforward, thanks to the use of modules.

4.3 Which version of the compiler and version of the MPI library did you use?

The Cray compiler was utilized. The following modules were loaded:

- PrgEnv-cray/8.3.3
- Cray-mpich/8.1.18

4.4 Did you use any tools to study the behaviour of your code?

We utilized timers already implemented within our solver.

5 Execution step

5.1 How is the program launched?

The executable file was launched using scripts.

5.2 Difficulties met to launch the code, if any, and how they were tackled.

None.

6 Communication patterns

If you know which are the main communication patterns used in your code configuration, select the ones from the mentioned below:

- ☒ Few point to point communications
- ☒ Few collective communications
  - ☐ Barrier
  - ☒ Reduction
  - ☒ Broadcast
  - ☐ Scatter/gather
  - ☐ All to all

7 Results of the scalability testing

7.1 Summary of the obtained results from the scalability testing

The results of the scalability tests were utilized to select the suitable/affordable size of the computational grid to be utilized to perform future production runs on a bio-inspired marine propeller (about 12 billion points). They were also utilized to estimate the computational cost of the planned computations (about 60 million core-hours). Tests were conducted to demonstrate both strong and weak scalings of the solver on LUMI-C. Performance was demonstrated very satisfactory, up to the selected size of the flow problem to be studied by means of future production runs.

7.2 Images or graphics showing results from the scalability testing

Figure 1 shows the strong scaling performance of the solver on a computational grid consisting of 11.6 billion points. The behaviour is almost linear up to 64 nodes, corresponding on LUMI-C to 8,192 cores. Therefore, we selected this setup for future production runs.

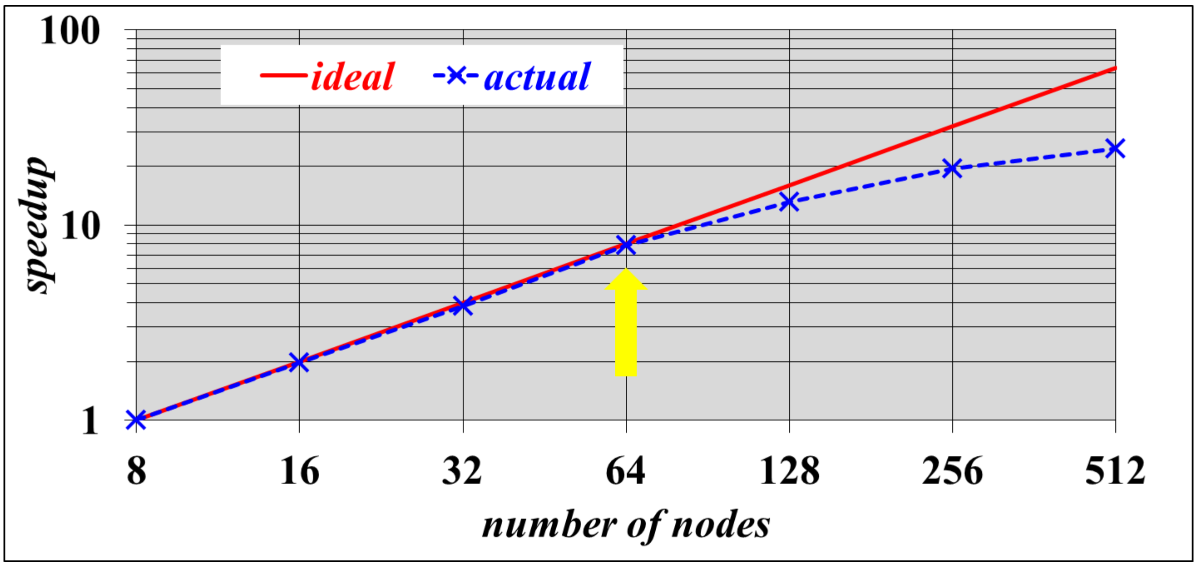

Figure 1 Results of strong scaling tests on LUMI-C

Weak scaling is also demonstrated in figure 2, where the yellow column deals with the reference case of a cylindrical grid consisting of 11.6 billion points, using 64 nodes of LUMI-C. All timings were normalized considering this configuration. Also weak scaling is very satisfactory.

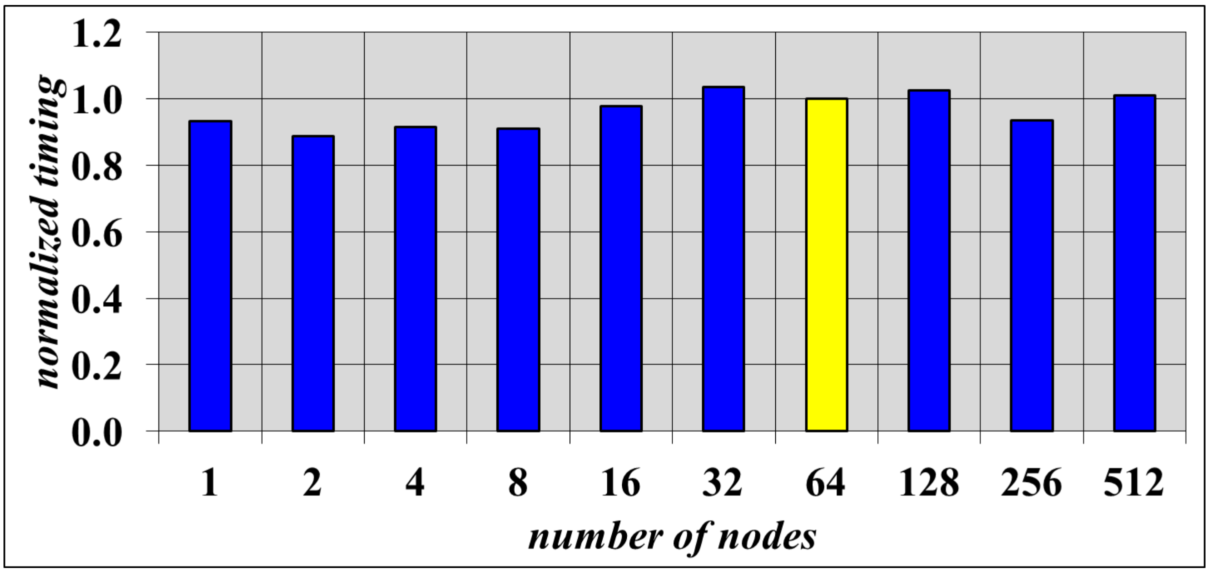

Figure 2 Results of weak scaling tests on LUMI-C

7.3 Data to deploy scalability curves

A) Some typical user test cases

Number of cores	Wall clock time (s/step)	Speed-up vs the first one	Number of Nodes	Number of process
4,096	23.15	1.00	32	4,096
8,192	11.30	2.05	64	8,192
16,384	6.78	3.42	128	16,384

B) Strong scaling curve

Number of cores	Wall clock time (s/step)	Speed-up vs the first one	Number of Nodes	Number of process
1,024	88.65	1.00	8	1,024
2,048	44.84	1.98	16	2,048
4,096	23.15	3.83	32	4,096
8,192	**11.30**	**7.85**	**64**	**8,192**
16,384	6.78	13.08	128	16,384
32,768	4.56	19.45	256	32,768
65,536	3.59	24.67	512	65,536

C) Weak scaling curve

Number of cores	Wall clock time (s/step)	Speed-up vs the first one	Number of Nodes	Number of process
128	10.52	1.00	1	128
256	10.03	1.05	2	256
512	10.33	1.02	4	512
1,024	10.28	1.02	8	1,024
2,048	11.06	0.95	16	2,048
4,096	11.70	0.90	32	4,096
8,192	**11.30**	**0.93**	**64**	**8,192**
16,384	11.57	0.91	128	16,384
32,768	10.56	1.00	256	32,768
65,536	11.43	0.92	512	65,536

7.4 Publications or reports regarding the scalability testing

8 Results on Input/Output

8.1 Size of the data and/or the number of files

I/O operations were conducted only to verify the ability of the subroutines implemented within the solver to properly generate checkpoint files of the solution (4 files, having a size of 87GB each). These tests were successful. Checkpoint files were immediately removed. A single diagnostic file having a size of a few kB was generated by each test, storing information for profiling.

8.2 Please, let us know if you used some MPI-IO features

I/O operations were conducted using calls to parallel HDF5 libraries.

9 Main results

We demonstrated excellent strong and weak scaling performance of our in-house fluid dynamic solver on LUMI-C. The present project was also useful to select the most appropriate size of the flow problem to be proposed in the framework of EuroHPC JU Regular Access as well as to estimate the amount of resources to ask for production runs (about 60 million core-hours). Our plan is to simulate a bio-inspired propeller on a cylindrical grid consisting of about 12 billion nodes, performing computations on 64 nodes of LUMI-C.

10 Feedback and technical deployment

10.1 Feedback on the centres/EuroHPC JU mechanism

We are very satisfied. The response to our application to EuroHPC JU Benchmark Access for computational resources on LUMI-C, to perform scaling tests, was very fast, within a few days from the cut-off date. Having already verified the suitability of our solver to LUMI-C, we have plenty of time for the preparation of our proposal for Regular Access to LUMI-C.

10.2 Explanation of how the computer time was used compared with the work plan presented in the proposal. Justification of discrepancies, especially if the computer time was not completely used.

No major issues were encountered when porting our solver on LUMI-C. Therefore, we were able to complete all required scaling tests in advance, within the first month of the allocation.

10.3 Please, let us know if you plan to apply for EuroHPC JU Regular Access in the future? If not, explain us why.

Yes, we applied indeed for Benchmark Access with the purpose of preparing our next proposal for Regular Access. This allowed us to produce the information on scaling performance required by the Regular Access policy, to set up our future production runs and to estimate the computational resources on LUMI-C we need to ask for.