**RESPONSE TO REVIEWER @thomasgillis – RHEA: an open-source Reproducible Hybrid-architecture flow solver Engineered for Academia #4637**

The authors would like to thank the reviewer for the valuable comments, suggestions and remarks. We have answered all of them below.

Comments and answers:

**1. The paper is 1900 words while the limit is at 1000, maybe you will want to reduce it. There are plenty of non-relevant details that could go away and you might want to tighten your english a bit more. I would definitely rewrite your sections summary, statement of the need, etc. to make them more explicit about what you do and others do or don't.**

Following your indications, we have substantially reduced the amount of text in the “Summary” and “Statement of need” sections. We have also simplified some of the sentences in the other sections to further reduce the total number of words in the document. In addition, we have further emphasized the differences between our solver and other accelerated open-source flow solvers.

**2. You claim to be open-source and reproducible, what makes the code more reproducible than others? What does it mean to you to be reproducible?**

We refer to “reproducible” when the data generated from a solver can be replicated by other people, either (i) using a different solver or (ii) by being able to rerun the simulation with the same solver and exact parameters/conditions. In our case, we focus on the latter, and consequently we plan to upload to the repository (in the tests folder) all the input and parameter files required to simulate the (most important) flow problems we publish in the literature. As it can be seen in the “tests” folder of the repository, we have already done that for 9 canonical flow problems. We have emphasized this idea by adding the following text at the end of “Statement of need”: “…, and (v) reproducibility through uploading to the repository the parameters and input files required to regenerate the data from simulations.”.

**3. Your relationships between N and Re should either be detailed or refer to a citation (and I have N ~ Re^3 in mind).**

We have rewritten the sentence to “In particular, the computational cost of studying wall turbulence, in terms of total number of grid points N, by means of direct numerical simulation (DNS) and/or wall-modeled large eddy simulation (LES) strategies scales with the Reynolds number as $N \sim Re^{37/14}$ and $N \sim Re$ [@Choi2012-A], respectively.”. The reference [@Choi2012-A] corresponds to: Choi & Moin. Grid-point requirements for large eddy simulation: Chapman's estimates revisited. Physics of Fluids, 24, 011702, 2012.

**4. Your figure 1 (left) is very unclear to me. "First, when using CPUs+GPUs with respect to only CPUs, the solver is accelerated approximately 2.5× and 5× on the Hybrid and BSC computers" where do you see that, I only see time and energy.**

Figure 1(left) shows the ratios of time-to-solution and energy-to-solution using the solver with CPU and CPU+GPU on two different computing platforms: hybrid (first two bars) and BSC (second set of bars). The red bars represent the ratio "time-to-solution using CPU+GPU/time-to-solution using CPU", whereas the blue bars represent the ratio "energy-to-solution using CPU+GPU/energy-to-solution using CPU". Focusing on the red bars (ratio of time-to-solution), it can be seen that on the Hybrid machine the solver using CPU+GPU takes approximately a fraction of 0.4 of the time it takes using only CPU, which corresponds to an acceleration of roughly $1.0/0.4 \approx 2.5$. Similarly, on BSC, the solver is accelerated by a factor of approximately $1.0/0.2 \approx 5$ when using CPU+GPU with respect to only CPU. In this regard, to facilitate the comprehension we have modified the text as "First, focusing on the time-to-solution when using CPUs+GPUs with respect to only CPUs, the solver is accelerated approximately 2.5x and 5x on the Hybrid and BSC computers.".

**5. Please add a weak analysis and some sense of the percentage of parallel regions in your program (can be numerically estimated from both the strong and weak scaling). The sentence "the solver presents nearly ideal speedup" is very subjective.**

Following these indications we have: (i) added weak scalability results in Figure 1 and discussed them in the text, (ii) quantified the number of grid points per processor at the maximum node count, (iii) modified the sentence "... the solver presents nearly ideal speedup …" to "... the solver presents similar speedups in terms of strong scalability when running on CPUs and CPUs+GPUs up to 32 nodes …". As a result, the paragraph discussing the computational performance results has been modified to "The computational performance of RHEA is depicted in Figure 1 by carrying out (i) time- & energy-to-solution, and (ii) strong scalability tests based on 100 time iterations of the 3-D turbulent channel flow configuration (different mesh) described in the next section. In this regard, to assess the portability of RHEA to different computing systems, the performance tests have been performed on (i) a local hybrid machine (results are referred to as Hybrid) and (ii) the Barcelona Supercomputer Center [@BSC] (results are indicated as BSC) without performing any particular tuning of the solver. The Hybrid machine is composed of a node with 1 AMD Ryzen 9 3900XT 12-core CPU and 2 NVIDIA Quadro RTX 4000 GPUs, while the BSC supercomputer contains a CPU-GPU cluster with 3 racks formed of 54 IBM POWER9 nodes, each containing 2 Witherspoons CPUs (20 cores of 3.1 GHz each), 4 Volta NVIDIA GPUs (16 GB each), and 6.4 TB of NVMe (Non-Volatile Memory). Four main observations can be inferred from the results. First, focusing on the time-to-solution when using CPUs+GPUs with respect to only CPUs, the solver is accelerated approximately 2.5x and 5x on the Hybrid and BSC computers. Second, running on CPUs+GPUs consume more power (Joules per second, i.e., Watts) than on CPUs. However, the energy-to-solution (Joules) is reduced by factors of roughly 1.3 and 2.5 on the Hybrid and BSC systems, respectively. Third, on BSC for a fixed-problem size, the solver presents similar speedups in terms of strong scalability when running on CPUs and CPUs+GPUs up to 32 nodes (640 cores and 128 GPUs) until communication overheads become important in relative value; at 32 nodes, each processor contains approximately 325000 grid points. Fourth, maintaining the same ratio of problem size to number of nodes, RHEA is able to preserve weak scalability efficiency above 90% on CPUs and CPUs+GPUs up to 32 nodes on BSC.”.

**6. Although 32 nodes is nice, the rank count is very low (<1000) and contrasts with your claim "mid-size workstations to the largest supercomputers worldwide". I think that any recent MPI implementations will have no problems to scale up to 600 ranks. Any chance you can get above 4k to present a more accurate picture?**

We used the maximum number of nodes available in the CPU+GPU cluster at the Barcelona Supercomputing Center available to perform scalability tests through a dedicated allocation. At maximum node count, the calculations utilized 640 cores + 128 CPUs, which we believe is relevant for most of the calculations we will plan to run in the coming years to study the flow physics of multiscale flow problems of our interest. However, the idea we wanted to highlight is that the results presented, in connection to the results from the Hybrid machine, were obtained without changing a line of code in the source files of the solver; only paths in the Makefile for compiling. In this regard, since our objective is to balance computational performance and portability, we have modified in the results section the following text “In this regard, to assess the portability of RHEA to different computing systems, the performance tests have been performed on (i) a local hybrid machine (results are referred to as Hybrid) and (ii) the Barcelona Supercomputer Center [@BSC] (results are indicated as BSC) without performing any particular tuning of the solver.”.

**7. It's difficult to verify your authorship as the contributions on gitlab come from the same user "ProjectRHEA" and you are not recognized as an author on gitlab. Any way to fix that?**

The name “ProjectRHEA” is a generic username Guillermo Oyarzun (third author of the manuscript) and Lluis Jofre (myself - first author of the manuscript) created for the GitLab project. Most of the time, we use that username to make changes to the code. As it can be seen in the “LICENSE” file of the project (MIT license created more than 2 years ago when the repository was initiated), the authorship of the repository belongs to Lluis Jofre Cruanyes (myself - first author of the paper) & Guillermo Oyarzun Altamirano (third author of the paper). From now on, we will make updates to the repository using our individual usernames.

**8. You don't have any automated documentation in the code for the most part of the classes I could visit. It makes the software very difficult to understand as well as difficult to maintain.**

We have incorporated Doxygen into the project as an automatic documentation generator. A new folder named “doxygen” is available in the main directory. The folder contains the configuration file “Doxyfile” and the folder “html” with the documentation of all the classes and functions in the “src” directory. The documentation can be rapidly updated by running “doxygen Doxyfile”.

**9. You don't have any CI/CD pipeline on gitlab.**

We have implemented a CI/CD pipeline to test the solver everytime a commit is made.

**10. I randomly checked your code and you sometimes have weird coding practices like using “namespace std;” in headers or enforcing a MPI_Barrier before stopping a timer.**

Following these indications, we have removed all the “namespace std;” in the header files.

The “MPI_Barrier” instructions are only used in the ParallelTimer class (ParallelTimer.cpp file) to synchronize the processes when analyzing the time taken by each function of the code. This is needed for analyzing the computational performance of MPI-based programs in “testing” mode. However, these barriers do not affect the performance of the code when running in “production” mode as the timers are turned off.

**RESPONSE TO REVIEWER @ctdegroot – RHEA: an open-source Reproducible Hybrid-architecture flow solver Engineered for Academia #4637**

The authors would like to thank the reviewer for the valuable comments, suggestions and remarks. We have answered all of them below.

Comments and answers:

**1. Installation instructions are lacking. I don't have YAML installed and there are no instructions or links about how to install it or what version I need. The instruction to "Modify Makefile (paths & flags) according to the computing system" is not great. This needs to be improved.**

We have updated the installation instructions in the README.md and Makefile files: (i) yaml-cpp version needed, (ii) HDF5 module compatibility with MPI library, (iii) listed variables to set in the Makefile, and (iv) provided in the Makefile examples for Ubuntu (Linux) and MAC (OS X).

Particular installation instructions will vary from system to system (e.g., linux workstation, MAC laptop, local cluster, supercomputer). However, the external dependencies required are minimal (C++ compiler, MPI library, yaml-cpp implementation and HDF5 module) and quite standard. For example, (i) the following commands can be used on Ubuntu (other options may also work): sudo apt install libopenmpi-dev libyaml-cpp-dev libhdf5-openmpi-dev, whereas (ii) these modules can be load on most supercomputers: module load openmpi yaml hdf5.

**2. There is no apparent documentation for example usage. There are some cases tucked into the "tests" directory but no instructions on what to do with them.**

The README.md file has been extended to incorporate instructions on how to run the tests:
EXAMPLE TESTS:
- Change directory to any of the examples provided in the tests folder
- Adapt Makefile to the computing system (or copy the Makefile previously modified)
- Compile main file (myRHEA.cpp) by executing: $ make
- Start test by running (4 CPUs by default): $ ./execute.sh

**3. I opened some files at random and did not see much documentation of the software API. For example, the class ManagerHDF, there is absolutely no documentation about what this does or how to use it. It does not seem to meet the requirements for "Functionality documentation" described above.**

We have incorporated Doxygen into the project as an automatic documentation generator. A new folder named "doxygen" is available in the main directory. The folder contains the configuration file “Doxyfile” and the folder “html” with the documentation of all the classes and functions in the “src” directory.

**4. Integration tests do not seem to be automated and there are no instructions how to run them. I don't see any unit tests to document functionality of various functions/classes.**

We have implemented a CI/CD pipeline to test the solver everytime a commit is made.

**5. There are no guidelines for how to contribute to the software, report issues, or seek support.**

We have incorporated guidelines for support, issues and contribution in the README.md file:
SUPPORT, ISSUES & CONTRIBUTION
- Documentation of the classes, functions and variables is available in doxygen/html
- Additional support is provided via the "Service Desk" of the project
- Issues can be reported utilizing the "Issues" functionality menu
- Contributions can be proposed, discussed and incorporated through "Merge requests"