COMBS: First Open-Source Based Benchmark Suite for Multi-physics Simulation Relevant HPC Research

Anthony Dowling[1], Frank Swiatowicz[2], Yu Liu[1(✉)], Alexander John Tolnai[3], and Fabian Herbert Engel[4]

[1] Clarkson University, Potsdam, NY 13699, USA
{dowlinah,yuliu}@clarkson.edu
[2] Vassar College, Poughkeepsie, NY 12604, USA
fxswiatowicz@vassar.edu
[3] Royal Melbourne Institute of Technology, Melbourne, VIC 3000, Australia
s3540645@student.rmit.edu.au
[4] Hasso Plattner Institute, 14482 Potsdam, Germany
fabian.engel@student.hpi.de

Abstract. Recent scientific computing increasingly relies on multi-scale multi-physics simulations to enhance predictive capabilities by replacing a suite of stand-alone simulation codes that independently simulate various physical phenomena. Inevitably, multi-physics simulation demands high performance computing (HPC) through advanced hardware and software accelerating due to its intensive computing workload and runtime communication needs. Thus, its research has become a hotspot across different disciplines. However, it is observed that most benchmarks used in the evaluation of corresponding work are through commercial or in-house codes. Then, the lack of accessible open-source multi-physics benchmark suites has presented a challenge in uniformly evaluating simulation performance across related disciplines. This work proposes the first open-source based benchmark suite with 12 selected benchmarks for research in multi-physics simulation, the Clarkson Open-Source Multi-physics Benchmark Suite (COMBS). Multiple metrics have been gathered for these benchmarks, such as instructions per second and memory usage. Also provided are build and benchmark scripts to improve usability. Additionally, their source codes and installation guides are available for downloading through a github repository built by the authors. The selected benchmarks are from key applications of multi-physics simulation and highly cited publications. It is believed that this benchmark suite will facilitate to harness the full potential of HPC research in the field of multi-physics simulation.

Keywords: Multi-physics simulation · High performance computing · Benchmark suite · Open-source

M. Qiu (Ed.): ICA3PP 2020, LNCS 12452, pp. 3–14, 2020.
https://doi.org/10.1007/978-3-030-60245-1_1

1 Introduction

Recent trends in scientific computing increasingly rely on multi-scale multi-physics computer simulations to enhance predictive capabilities by replacing conventional methods that are largely empirically based with a more scientifically based methodology. Through this approach, one addresses the issue of traditionally employing a suite of stand-alone codes that independently simulate various physical phenomena that were previously disconnected [1]. For example, coupled multi-physics codes have been developed worldwide to address the growing concerns of reactor performance and nuclear safety [1–5]. Also, multi-physics simulations are helping researchers make progress in finding effective cancer treatments through simulating and modeling the background dose of targeted alpha therapy (TAT) [6–8]. Moreover, multi-physics simulation plays an important role in semiconductor design [9] and aerospace engineering [10].

Multi-physics simulation requires capturing many physical interactions among complex phenomena through rigorous coupling at run-time, facilitating the need for high performance computing (HPC) through advanced hardware and software acceleration due to its massive computing workload and fast run-time communication needs. Thus, prior research has been done to boost the general software and hardware HPC computing environment for multi-physics simulations in recent years, which is briefly described in Sect. 2. However, the authors realize that one of the research obstacles in this domain is the lack of an open-source benchmark suite for research evaluation. The lack of accessible open-source multi-physics benchmark suites has presented a challenge in uniformly evaluating simulation performance across related disciplines.

To the best of our knowledge, this work proposes the first open-source benchmark suite for multi-physics simulation relevant HPC research, the Clarkson Open-Source Multi-physics Benchmark Suite (COMBS). These benchmarks have been installed and evaluated on an Ubuntu Linux server, e.g., instructions per second or memory usage. Also provided are build and benchmark scripts to improve usability. Additionally, their source code and installation guides are available for downloading through a Github repository built by the authors. The selected 12 benchmarks are from key applications of multi-physics simulations and highly cited publications, which will be detailed in Sect. 3. It is believed that this benchmark suite will facilitate to harness the full potential of the HPC research in the field of multi-physics simulation. The contributions of this paper are three-fold:

- Propose the first benchmark suite for multi-physics simulation relevant HPC research;
- Each benchmark is open-source, allowing for free access and modification as desired;
- All benchmarks have been installed and evaluated on an Ubuntu Linux server.

The rest of the paper is organized as follows. Section 2 describes the related work and motivation of this paper. Section 3 introduces the selected benchmarks

in detail, along with the benchmark selection criteria. The benchmark data and website are illustrated in Sect. 4. Lastly, the authors summarize the conclusion of this work and discuss planned future work in Sect. 6.

2 Related Work and Motivation

To satisfy the HPC needs of multi-physics simulation, many research projects have been done to adapt simulation codes, optimize the HPC based software framework, and explore novel computer architectures for HPC systems. One research focus is in exploring software coupling platforms, simulation codes (a.k.a., applications), algorithms with the objective of reaching a performance boost through the HPC systems, exchange run-time data effectively, and cost-effective transition. The other focus is in hardware design, e.g., investigating multi-physics simulation aware computer architectures and HPC systems. The objective of recent research is to improve the general software and hardware HPC computing environment for large-scale multi-physics simulations.

Many HPC technique based code coupling platforms have been developed; the capabilities and intended applications of each code differ, such as the Backbone [1], MOOSE [11], LIME [12], and SALOME [13]. The Backbone, developed at the Canadian Nuclear Laboratories (CNL), has the capability of synchronizing various reactor performance and safety analysis codes, which permits appropriate and efficient coupling at the time step level. Idaho National Laboratory (INL) proposed the Multi-physics Object Oriented Simulation Environment (MOOSE) [11], and a code matching MOOSE standard can "plug and play" into the environment. The Open Source Integration Platform for Numerical Simulation (SALOME) [13], funded by the French Energy Commission (CEA), is based on the model of distributed components as a distributed objects architecture. The Lightweight Integrating Multi-physics Environment (LIME) toolset [12], developed by Sandia National Laboratory (SNL), is used within the Consortium for Advanced Simulation of Light Water Reactors (CASL) hub and also supports coupling of multiple codes in other fields. Besides this platform based work, other research focuses on resource allocation and load balancing to achieve an effective run-time computing environment for multi-physics simulations [14,15].

Compared to general HPC applications, multi-physics simulations have their own unique characteristics. For example, the simulation codes involved in a simulation may request intensive run-time data to be exchanged by them. Also, some applications (e.g., Monte Carlo (MC)) with long consequential computing-intensive codes may run faster on multi-core CPUs, which offer high degrees of instruction level parallelism (ILP). Other codes (e.g., Computational Fluid Dynamics (CFD)) might have a high degree of parallelism, which could make use of the hundreds of simple and effective cores of general purpose GPUs. Thus, much work has been done to satisfy these unique needs through designing advanced computer architectures or HPC systems, such as heterogeneous design, 3D architecture, etc. [16].

However, it is observed that their evaluation is based on either commercial/in-house simulation codes (a.k.a., applications), or limited applications in specific

areas. Research on designing a benchmark suite for multi-physics simulations is inspired by the lack of available open-source suites that measure performance evaluation. Some of the available benchmarks are locked behind licenses and use in-house codes that make easy comparisons across different simulations challenging. Benchmark suites have been created to measure the performance of multi-physics simulations but suffer from certain flaws. No open source benchmark suite has ever been proposed to facilitate the relevant evaluation work specific to multi-physics simulation.

For example, COMSOL and other commercial packages provide benchmarks to test simulations but the codes are not open-source, and require a license to use. In [1,6,7], in-house simulation codes (e.g., Element Loss-Of-Coolant Accidents (ELOCA), Canadian Algorithm for Thermalhydraulic Network Analysis (CATHENA)) are used, which are confidential and not accessible by researchers outside CNL. In [14], the authors only mentioned the benchmark is a MC simulation used as a bigJob, while [15] used 7 problems named P1 to P7. Both [14] and [15] work on resource allocation algorithms for multi-physics simulations.

Since very few of these works provide detailed information regarding their benchmarks, **is it possible for us to evaluate their advantages and disadvantages? Is it fair if one of them declares performance improvement over the other?** In addition, general purpose open-source benchmark suites like CORAL-2 [17] are just a collection and still require users to individually install, build, and run each benchmark in the suite. The CORAL-2 benchmarks also include wide variations in lines of code and other attributes like uneven testing of CPU compared to GPU.

Thus, it is very difficult for researchers to duplicate the achievements of previous publications. In addition, evaluating performance between new proposed systems and legacy ones is almost impossible, since the evaluations are based on different benchmarks. Thus, the authors believe that this situation becomes an obstacle for future research regarding multi-physics simulations. Thus, it is necessary to propose an open-source based benchmark suite to address this concern. The authors believe that the open-source nature of such a suite could guarantee the access of benchmarks and build a general foundation for relevant research.

3 Benchmark Suite

Through the compilation of 12 benchmarks that measure performance indicative of multiple physics algorithm designs, research in the field will benefit from an open-source suite that is modifiable and free to use. The benchmark codes have been tested to successfully build on an Ubuntu 18.04 server, allowing for the codes to be ran in a stable, easily configurable environment.

3.1 Selection Criteria

In order to construct a benchmark suite that would provide adequate coverage and measure desired performance, potential benchmarks have to meet certain criteria, as follows:

– open-source;
– able to be built and compiled with few additional external libraries;
– written in C or C++ as to allow for the use of a single compiler;
– measures performance typical of an aspect of multi-physics simulations (e.g., tightly coupled codes).

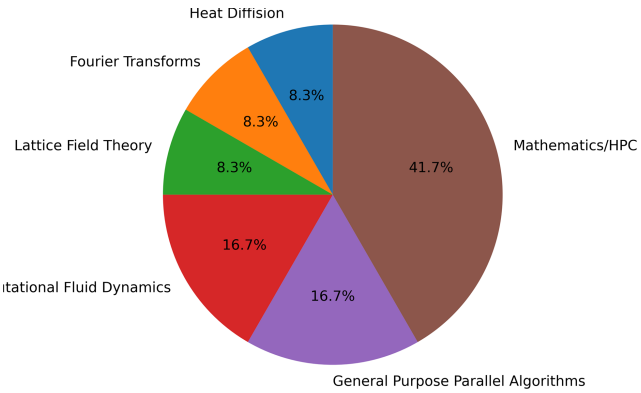

Fig. 1. A pie chart showing the different multi-physics-related benchmarks that are present in the suite. Note the relatively even distribution among selected topics.

The last point was of particular importance, as it is difficult to determine the most salient aspects of a field with limited background experience in multi-physics algorithm design. Upon consulting literature in the field, it was determined that finding algorithms that solve partial differential equations (PDEs) was necessary, as multi-physics simulation solution times depend on how quickly PDEs can be solved. Secondary characteristics used for the selection of benchmarks included their relatedness to the field in terms of the problem they solved or their presence in licensed benchmark suites. In this work, queries were conducted primarily through GitHub.

3.2 Benchmarks

The benchmark suite proposed in this paper is named the Clarkson Open-Source Multi-physics Benchmark Suite (COMBS). A total 12 benchmarks are divided into 2 groups, including 8 key application/framework benchmarks, and 4 supplemental benchmarks. The distribution of the selected 12 benchmarks are shown in Fig. 1.

The 8 Selected Key Application/platform Benchmarks. These 8 benchmarks consist of 8 applications. Each benchmark is open-source, allowing for free access and modification as desired. Each benchmark consists of algorithm(s) that

solve problems related to the field. As an example, one of the selected benchmarks is a solver for the Kuramoto-Sivashinsky (KS) equation. The KS equation is one of the principal equations in connecting PDEs and dynamical systems. Since multi-physics simulations rely heavily on the tight coupling between dynamic algorithms, this benchmark appears to be reasonable to test baseline performance, at least on a much simpler, one dimensional problem. The KS equation is a highly cited problem in the field dating back over 30 years and being a PDE problem, fits directly in to multi-physics, since such systems depend on the how quickly a PDE system can be solved [18]. Similar reasoning is given for the other benchmarks, which are hosted on a GitHub repository.

The 8 application type benchmarks can be coupled through the distributed code coupling platforms, such as CNL's Backbone [1], CEA's SALOME [13], SNL's LIME [12], etc. The 8 key benchmarks are described in the following.

Benchmark - 2D Heat: This benchmark compares the utility of MPI, OpenMP, and serial implementations of the 2d-heat benchmark [19]. This work is related to HPC applications as shown in related papers that attempt to parallelize the 2D heat equation to improve performance [20]. The tight coupling of a heat distribution algorithm makes it an obvious candidate for a multi-physics benchmark suite.

Benchmark - Advection-Diffusion Equation: Advection-diffusion equations [21] are used in the field of CFD to measure phenomena like the time evolution of chemical or biological species in a flowing medium such as water or air. Since such an evolution can be modeled with PDEs, an algorithm that models a simpler, one dimensional aspect of advection-diffusion can be useful in measuring simulation performance. This benchmark was adapted from open-source codes at [22].

Benchmark - Fidibench: FiDiBench is a finite difference suite of codes that can be used to benchmark hardware performance on HPC and other systems. The code examples are small enough to be well understood, typically averaging a few hundred lines of code, but are also relevant to scientific computing, which often involves nearest neighbor communication. FiDiBench can be used to compare the execution speed obtained by implementing a given algorithm in different languages (e.g., C++ vs Python vs Julia). This benchmark was adapted from open-source codes at [23].

Benchmark - High Performance Conjugate Gradient (HPCG): HPCG [24] is a software package that performs a fixed number of multigrid preconditioned (using a symmetric Gauss-Seidel smoother) conjugate gradient (PCG) iterations using double precision floating point values. The HPCG rating is a weighted GFLOP/s (billion floating operations per second) value that is composed of the operations performed in the PCG iteration phase over the time taken. The overhead time of problem construction and any modifications to improve performance are divided by 500 iterations (the amortization weight) and added to the run-time.

Benchmark - KS-PDE: The Kuramoto-Sivashinsky (KS) equation [18] is one of the principal equations in connecting partial differential equations (PDEs) and dynamical systems. Since multi-physics simulations rely heavily on the tight coupling between dynamic algorithms, this benchmark appears to be reasonable to test baseline performance, at least on a much simpler, one dimensional problem. The KS equation is a highly cited problem in the field dating back over 30 years and being a PDE problem, fits directly in to multi-physics, since such systems depend on the how quickly a PDE system solved.

The code for this benchmark is adapted from a recent open-source work [25], which is in an attempt to compare run-time performance between different programming languages, with the goal of legitimizing Julia as having the potential to be among the fastest languages if used correctly. Additionally, the benchmark can be adapted to test the performance of just the C code.

Benchmark - 2D Lid-Driven Cavity: This is a concise finite difference method based code for solving the Navier-Stokes equation in a 2D Lid-driven cavity. The problem is to move fluid in a box from one corner to another, measuring the fluid's velocity. The Navier-Stokes equation is commonly used to solve CFD relevant problems and then couple with other codes in many multi-physics simulation scenarios.

Benchmark - Phase Retrieval: The mathematical and algorithmic aspects of the phase retrieval problem have received considerable attention. In crystallography, a principal application in this area, the signal to be recovered is periodic and comprised of atomic distributions arranged homogeneously in the unit cell of the crystal. The crystallographic problem is both the leading application and one of the hardest forms of phase retrieval [26]. This benchmark is used to solve phase retrieval, which uses techniques such as 3D Fourier Transforms that are applicable to multi-physics simulation designs. This benchmark is adapted from a recent open-source implementation at [27].

Benchmark - SOMBRERO: In physics, lattice field theory is the study of lattice models of quantum field theory, i.e., of field theory on a spacetime that has been discretized onto a lattice. Lattice field theory is directly applicable to high performance computing and adjacent multi-physics simulations. This benchmark is adapted from open-source codes at [28]. SOMBRERO runs 6 sub-benchmarks, each representing a theory under active study by the lattice field theory community. For the purposes of benchmarking, the models vary only in the amount of data communicated and the number of floating point operation required.

The 4 Supplement Benchmarks of HPC Mathematics/Algorithms. Along with the 8 benchmarks selected from the key areas involved in the multi-physics simulation, the authors also chose 4 benchmarks regarding general HPC mathematics and algorithms, e.g., Monte Carlo, MPI matrix multiplication, Gaussian Blur, Radix Sort, etc. These benchmarks are a good complement to the key benchmarks for evaluating the basic performance and optimization of the

proposed new software and hardware systems, such as inter-node communication latency, shared last level cache (LLC) performance, etc.

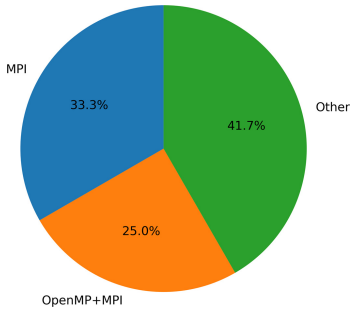

Fig. 2. HPC Techniques Coverage

3.3 HPC Coverage Analysis

Figure 2 shows the coverage of HPC techniques of the 10 key application/framework benchmarks. It is observed that about 70% of them utilize Message Passing Interface (MPI) [29] and OpenMP [30], which are the most popular HPC techniques used in the field of multi-physics simulation. Although 30% of them are based on other techniques, they can be easily coupled through the distribution style software framework like Backbone, SALOME and LIME.

4 Benchmark Repository on Github

A Github repository has been designed to include the source codes of all benchmarks, the installation guide (tested on Ubuntu Linux 18.04). This website is shown in Fig. 3, and the link is at [31]. Users can download the source codes of all benchmarks through the git utility, and then follow the guide to install the dependencies for the benchmarks and run them.

5 Characteristics of the Benchmark Suite

By building and running the benchmark suite using the provided scripts in the github page built by authors, it is possible and convenient to create an overview of all the benchmarks with multiple metrics as the characteristics of this benchmark suite. Table 1 shows this characteristics of COMBS for selected benchmarks when formatted as a table. This run of the benchmarks was gathered on a Dell Precision 7920 Tower server with two Intel Xeon Silver 4110 16 3 GHz processors and 32 GB of RAM. The server also has two GPUs: an NVIDIA NVS 310 and

Table 1. Example Output from the Benchmark Suite showing selected Benchmarks

Benchmark	Time (s)	Instruction count (Millions)	Max memory usage (kB)	MIPS
2d-heat	12.628	62091.213	10400	4917
FourierBenchmarks	85.937	288618.737	53272	3358
advection-diffusion	9.638	170011.780	10360	17640
fidibench	43.565	50.128	178028	1.151
hpcg	99.667	222584.356	971244	2233
ks-pde	24.375	174739.281	21392	7169
lid-driven-cavity	3.553	31345.988	6492	8822
matrix-mpi	8.541	46539.884	535608	5449
monte-carlo	116.524	233543.444	1506492	2004
phase-retrieval-benchmarks	4.141	26685.456	15888	6444
radix_sort	4.968	37136.804	16052	7475
sombrero	59.478	8.747	22124	0.147

an NVIDIA TITAN Xp. The server is running Ubuntu 18.04.3 LTS x86_64 with Linux kernel 5.0.0-32-generic.

These metrics were gathered using tools from the valgrind toolset, and the timing functionality of the Linux time command [32]. First, the `time` command is used to time the wall-clock execution time of the program. This gives a feel for how long the program takes to run. Then, maximum memory usage is gathered using the `massif` [33] tool in the valgrind toolchain. This measurement shows the space complexity of the program as it is running, along with any extra memory that might be used by the program while it is in operation. Lastly, the `callgrind` [34] tool of valgrind was used to gather the instruction count of the program. This, along with the timing information and size of a program gives a way to see if a program may spend a lot of time in a loop, or if it is using more complex instructions.

These metrics were chosen because they are similar to the metrics used in [35], another research-oriented benchmark suite. Also, a comparison of the features of COMBS with CORAL-2 [17] mentioned in Sect. 2 is given in Table 2, which indicates the difference between the two open source benchmark suites. Furthermore, as described above, they give a method for determining certain platform-specific characteristics of a program, such as whether a long-running program simply spends a large amount of time in loops, or if a program is compiled to use complex instructions. Moreover, the memory usage measurement gives a method to

Table 2. Features of COMBS compared to CORAL-2

	COMBS	CORAL-2 [17]
Focus	Multi-physics Simulation	General Purpose
Compilation	Scripts to compile all	Compile each individually
Running	Automatic Run	Run each individually
Benchmark Size	Similar Size	Wide Variations

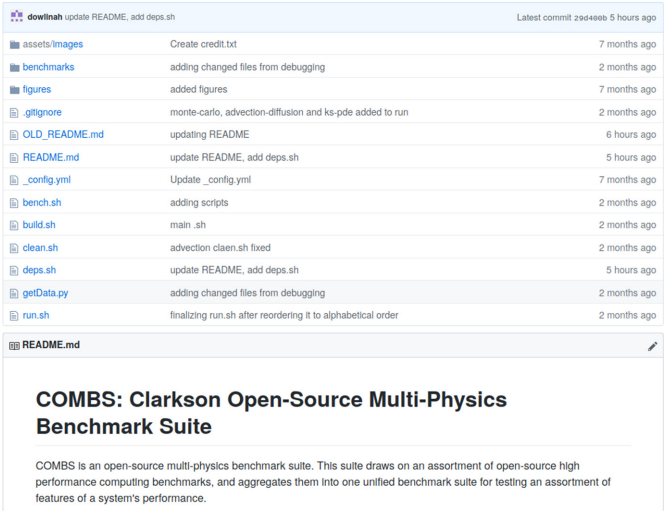

Fig. 3. Github Repository of Clarkson Open-Source Multi-physics Benchmark Suite (COMBS).

determine if a program may be accessing memory to a large degree; as high memory use would typically indicate high memory access. The authors believe these metrics are necessary for researchers to determine and adapt this benchmark suite based on their research purpose in the field of multi-physics simulation.

6 Conclusion and Future Work

The lack of accessible open-source multi-physics benchmark suites has presented a challenge in uniformly evaluating simulation performance across related disciplines. Through the compilation of 12 benchmarks that measure performance indicative of multiple physics algorithm designs, research in the field will benefit from an open-source suite that is modifiable and free to use. A dedicated Github repository is setup to host this benchmark suite, where users will have access to all documentation for each benchmark along with building scripts to facilitate easy use of the suite. Also provided are build and benchmark scripts to automatically perform the aforementioned steps. In the future, the authors would like to have the suite build and compile with a simulated computer architecture. Also, GPU acceleration-aware benchmarks will be selected and evaluated.

Acknowledgments. This work is partially funded by NSF Award OAC-1852102.

1 Artifact Description Appendix

A Github website [31] has been designed to include the source codes of all benchmarks and the installation guide (tested on Ubuntu Linux 18.04). Users can

download the source codes of all benchmarks through the command below, and then follow the installation guide of each benchmark to install and test them.

References

1. Liu, Y., Nishimura, M., Seydaliev, M., Piro, M.: Backbone: a multi-physics framework for coupling nuclear codes based on CORBA and MPI. Nucl. Eng. Radiat. Sci. (2017)
2. Gouja, I., Avramova, M., Rubin, A.: Development and optimization of coupling interfaces between reactor core neutronics and thermal-hydraulic codes. In: The International Conference on Advances in Reactor Physics to Power the Nuclear Renaissance (2010)
3. Gomez-Torres, A.M., Sanchez-Espinoza, V., Ivanov, K., Macian-Juan, R.: DYNSUB: a high fidelity coupled code system for the evaluation of local safety parameters-part I: development, implementation and verification. Ann. Nucl. Energy **48**, 108–122 (2012)
4. Sanchez, V., Al-Hamry, A.: Development of a coupling scheme between MCNP and COBRA-TF for the prediction of the pin power of a PWR fuel assembly. In: The International Conference on Mathematics, Computational Methods and Reactor Physics (2009)
5. Chen, Z., Chen, X.-N., Rineiski, A., Zhao, P., Chen, H.: Coupling a CFD code with neutron kinetics and pin thermal models for nuclear reactor safety analyses. Ann. Nucl. Energy **83**, 41–49 (2015)
6. Tao, X., Liu, Yu., Liu, T., Li, G., Aydemir, N.: Multiphysics modelling of background dose by systemic targeted alpha therapy. Med. Imaging Radiat. Sci. (2018). https://doi.org/10.1016/j.jmir.2018.06.002
7. Xu, T., Liu, T., Li, G., Dugal, C., Li, Y.: Microdosimetric and biokinetic modelling of alpha-immuno-conjugate transport in endothelial cells. J. Med. Imaging Radiat. Sci. **50**, S1–S2 (2019)
8. Xu, T., et al.: Technical note: the development of a multi-physics simulation tool to estimate the background dose by systemic targeted alpha therapy. Med. Phys. (2020)
9. Xiao, H.: A multi-physics approach to the co-design of 3D multi-core processors. Ph.D. dissertation (2018)
10. Errera, M., et al.: Multi-physics coupling approaches for aerospace numerical simulations. J. Aerosp. Lab (2011)
11. Schmidt, R., Hooper, R., Belcourt, N., Pawlowski, R.: MOOSE: a parallel computational framework for coupled systems of nonlinear equations. Nucl. Eng. Des. **239**(10), 1768–1778 (2009)
12. Schmidt, R., Belcourt, N., Hooper, R., Pawlowski, R.: An introduction to lime 1.0 and its use in coupling codes for multiphysics simulations. Sandia Report, SAND2011-8524 (2011)
13. SALOME official webpage (2019)
14. Ko, S.-H., Kim, N., Kim, J., Thota, A., Jha, S.: Efficient runtime environment for coupled multi-physics simulations: dynamic resource allocation and load-balancing. In: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (2010)

15. Sfika, N., Korfiati, A., Alexakos, C., Likothanassis, S., Daloukas, K., Tsompanopoulou, P.: Dynamic cloud resources allocation on multidomain/multiphysics problems. In: 3rd International Conference on Future Internet of Things and Cloud (2015)
16. Hermann, E., Raffin, B., Faure, F., Gautier, T., Allard, J.: Multi-GPU and multi-CPU parallelization for interactive physics simulations. In: D'Ambra, P., Guarracino, M., Talia, D. (eds.) Euro-Par 2010. LNCS, vol. 6272, pp. 235–246. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15291-7_23
17. CORAL-2 Benchmarks (2019). https://asc.llnl.gov/coral-2-benchmarks/
18. Hyman, J.M., Nicolaenko, B.: The Kuramoto-Sivashinsky equation: a bridge between PDE's and dynamical systems. Physica D: Nonlinear Phenomena **18**, 113–126 (1986)
19. 2D Heat Benchmark Source Codes (2011)
20. Horak, V., Gruber, P.: Multi-physics coupling approaches for aerospace numerical simulations. Parallel Numerics (2005)
21. Hundsdorfer, W.H., Verwer, J.G.: Numerical solution of time-dependent advection-diffusion-reaction equations. Parallel Numerics (2011)
22. Advection-Diffusion Equation Benchmark Source Codes (2017). https://github.com/antoine-levitt/benchmark_heat
23. Fidibench Benchmark Source Codes (2019). https://github.com/pletzer/fidibench
24. HPCG Benchmark Website (2019). http://www.hpcg-benchmark.org/
25. KS-PDE Benchmark Source Codes (2018). https://github.com/johnfgibson/julia-pde-benchmark
26. TAMIR BENDORY VEIT ELSER, TI-YEN LAN. Benchmark problems for phase retrieval (2017)
27. Phase Retrieval Benchmark Source Codes (2019). https://github.com/veitelser/phase-retrieval-benchmarks
28. Sombrero Benchmark Source Codes (2019). https://github.com/sa2c/sombrero
29. OpenMPI (2019). https://www.open-mpi.org/
30. OpenMP (2019). https://www.openmp.org/
31. COMBS Github (2020). https://github.com/dowlinah/COMBS
32. Nethercote, N., Seward, J.: Valgrind: a framework for heavyweight dynamic binary instrumentation. ACM SIGPLAN Not. **42**(6), 89–100 (2007)
33. Massif: a heap profiler (2020). https://valgrind.org/docs/manual/ms-manual.html
34. Callgrind: a call-graph generating cache and branch prediction profiler (2020). http://valgrind.org/docs/manual/cl-manual.html
35. Bienia, C., Kumar, S., Singh, J.P., Li, K.: The parsec benchmark suite: characterization and architectural implications. In: Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, pp. 72–81 (2008)