14th World Congress on Computational Mechanics (WCCM)
ECCOMAS Congress 2020
Virtual Congress: 11-–15 January 2021
F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

A GPU ACCELERATED FRAMEWORK FOR PARTITIONED SOLUTION OF FLUID-STRUCTURE INTERACTION PROBLEMS

Amin Totounferoush[1], Alireza Naseri[2], Jorge Chiva[2], Assensi Oliva[2] AND Miriam Mehl [1]

[1] University of Stuttgart, Institute for Parallel and Distributed Systems (IPVS)
Universitätsstraße 38, D-70569 Stuttgart, Germany
{amin.totounferoush, miriam.mehl}@ipvs.uni-stuttgart.de

[2] Universitat Politécnica de Catalunya - BarcelonaTech (UPC), Heat and Mass Transfer Technological Center (CTTC)
ESEIAAT, C/ Colom 11, 08222 Terrassa (Barcelona), Spain
{anaseri, jordic, oliva}@cttc.upc.edu

Key words: Hybrid CPU-GPU architectures, partitioned simulation, fluid-structure interaction

Abstract. We present a GPU-accelerated solver for the partitioned solution of fluid-structure interaction (FSI) problems. Independent scalable fluid and structure solvers are coupled by a library which handles the inter-code data communication, mapping and equation coupling. A coupling strategy is incorporated which allows accelerating expensive components of the coupled framework by offloading them to GPUs. To prove the efficiency of the proposed coupling strategy in conjunction with the offloading scheme, we present a numerical performance analysis for a complex test case in the filed of biomedical engineering. The numerical experiments demonstrate an excellent speed-up in the accelerated kernels (up to 133 times) which results in 6 to 8 times faster overall simulations. In addition, we observed a very good reduction in total simulation time by increasing the exploited compute nodes up to 8 (complete machine capacity).

1 Introduction

We present a partitioned solver which is capable of solving fluid-structure-interaction (FSI) problems exploiting both CPUs and GPUs efficiently. We follow a partitioned approach [1, 2] which decomposes the domain into smaller subdomains (in our case, fluid and structure subdomains) to be discretized and solved individually. Separate solvers and numerical settings are used for each subdomain along with a coupling technique to account for their interaction. This provides the opportunity to employ previously developed single-physics solvers which reduces software development cost. In addition, the most adapted and validated numerical methods for each sub-problem can be used, which considerably increases the reliability of the simulations. As a big advantage, the partitioned approach allows exploiting different hardware for each solver. This enables us to exploit heterogeneous CPU-GPU architectures by offloading the most expensive kernels of the coupled solver to GPUs.

Several partitioned methods for FSI simulation on parallel compute architectures have recently been presented in literature. As an example for CPU-only exploiting solvers, Hewitt et al. [3] presented a multi-code coupled framework using OpenFOAM for the fluid and ParaFEM for the structure. The solver is

shown to scale well on 1,536 cores for a coupled FSI problem. Naseri et al. [4] developed a scalable and efficient simulation environment for the partitioned solution of FSI problems. This framework employed two instances of TermoFluids [5] to solve the fluid and the solid problem and the preCICE coupling library [6] to handle inter-solver data communication, boundary data mapping and equation coupling. The strong scalability measurements proved high efficiency (83%) on up to 10,080 CPU cores. In addition, there are several hybrid architecture investigations for solving FSI problems in literature. For instance, Jiang et al. [7] presented a GPU-accelerated solver which follows a multi-code coupling strategy for the solution of FSI problems in the filed of biomechanics. A lattice Boltzmann solver was used for the incompressible fluid flow along with an explicit finite element solver for the solid domain. An Aitken relaxation was employed to improve the stability of the fixed point iterations. The numerical performance analysis showed that the GPU acceleration (using a single Nvidia GeForce 108Ti GPU) results in 18.44 times faster simulation compared to CPU-only simulation (on a Xeon E5-1620v4 CPU) for an FSI benchmark problem.

In the current study, we extend the solver presented in [4], to exploit hybrid architectures (CPU-GPU) efficiently. A coupling strategy is incorporated that allows using GPUs to accelerate the expensive kernels of our FSI coupling procedure, minimizing the total run–time while preserving the simulation accuracy. The presented setup can be applied to many other surface-coupled multi-physics problems such as conjugate heat-transfer, flow-acoustics coupling, or porous-media-free-flow coupling.

The remainder of this article is organized as follows. In Section II, we briefly introduce the governing equations and numerical methods for fluid and structure domain, as well as the coupling conditions and convergence acceleration techniques. Section III explains how the hybrid CPU-GPU architectures are exploited to accelerate the expensive kernels of the coupled solver. To show the efficiency of the proposed setup, we present the numerical performance analysis for a challenging FSI test case in Section IV. Finally, Section V summarizes and concludes the article.

2 Governing Equations and Numerical Methods

This section explains the governing equations for the fluid and the solid solver along with the numerical methods for solving them. In addition, the coupling conditions on the common interface are presented as well as the iterative procedure to achieve a converged solution at the common interface. In this work, we refer to the fluid and the structure domain as $\Omega_f(t) \subset \mathbb{R}^3$ for all t in $(0,T)$ and $\Omega_s(t) \subset \mathbb{R}^3$ for all t in $(0,T)$, respectively, where $t \in (0,T)$ represents time. The fluid-structure interface is the common boundary of the domains, denoted by $\Gamma(t) = \partial\Omega_f(t) \cap \partial\Omega_s(t)$.

2.1 Fluid Solver

2.1.1 Governing Equations

The Navier-Stokes equations are chosen to mathematically describe the unsteady flow of an incompressible viscous fluid. An Arbitrary Lagrangian-Eulerian (ALE) formulation together with a conforming mesh technique are employed to solve the fluid flow in a moving domain. The ALE formulation of the Navier-Stokes equations in a moving domain is given by

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{c} \cdot \nabla \mathbf{u} = \frac{1}{\rho_f} \nabla \cdot \sigma_f, \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2}$$

where $\mathbf{u}$ denotes the fluid velocity and ρ_f the fluid density. The ALE convective velocity is shown by vector $\mathbf{c} = \mathbf{u} - \mathbf{w}$, which is the fluid velocity relative to a domain moving with a velocity $\mathbf{w}$. The stress tensor σ_f for an incompressible Newtonian fluid is defined as

$$\sigma_f = -p\mathbf{I} + \mu_f(\nabla \mathbf{u} + \nabla \mathbf{u}^T), \tag{3}$$

where p is the fluid pressure, $\mathbf{I}$ the unit tensor, and μ_f the dynamic viscosity of the fluid.

2.1.2 Numerical Methods

An explicit time advancement method along with a fractional-step projection is used for the solution of the velocity-pressure coupling of the momentum equation. The following three step solution of the fluid governing equations must be followed from time step n to $n+1$

$$\begin{aligned} \mathbf{u}^p = \mathbf{u}^n - \Delta t[\frac{3}{2}(\mathbf{c}^n \cdot \nabla \mathbf{u}^n - \frac{\mu_f}{\rho_f}\Delta \mathbf{u}^n) - \\ \frac{1}{2}(\mathbf{c}^{n-1} \cdot \nabla \mathbf{u}^{n-1} - \frac{\mu_f}{\rho_f}\Delta \mathbf{u}^{n-1})], \end{aligned} \tag{4}$$

$$\frac{\Delta t}{\rho_{\mathrm{f}}}\Delta p^{\mathrm{n+1}} = \nabla \cdot \mathbf{u}^{\mathrm{p}}, \tag{5}$$

$$\mathbf{u}^{\mathrm{n+1}} = \mathbf{u}^{\mathrm{p}} - \frac{\Delta t}{\rho_{\mathrm{f}}}\nabla p^{\mathrm{n+1}} \tag{6}$$

in Ω_f^{n+1}, where Δt is the time increment and $\mathbf{u}^p$ is a predicted velocity field which does not satisfy the incompressibility condition (Eq. (2)).

The fluid equations are discretized in space using a finite-volume method on a collocated unstructured mesh with second-order symmetry-preserving schemes. These schemes conserve the mass, momentum and kinetic energy of the flow at the discrete level which is crucial in turbulent flow simulations [8, 9]. For more details please consult [4].

2.2 Structural Solver

2.2.1 Governing Equations

The conservation laws of mass and momentum govern the structural domain. The Lagrangian form is given by

$$\rho_s^0 = \rho_s J, \tag{7}$$

$$\frac{\partial}{\partial t}\left(\rho_s \frac{\partial \mathbf{d}}{\partial t}\right) = \nabla \cdot \sigma_s, \tag{8}$$

where the superscript 0 refers to the undeformed configuration of the body, ρ_s is the structural density and $\mathbf{d}$ is the displacement from the reference configuration. The hyper-elastic Saint Venant-Kirchhoff constitutive model relates the Cauchy stress tensor σ_s to the displacement field:

$$\sigma_s = \frac{\mathbf{B}}{2J}[2\mu_s(\mathbf{B}-\mathbf{I}) + \lambda_s \mathrm{tr}(\mathbf{B}-\mathbf{I})], \tag{9}$$

where $\mathbf{B}$ is the left Cauchy-Green deformation tensor $\mathbf{B} = \mathbf{F}\cdot\mathbf{F}^T$, and μ_s and λ_s are the Lamé's parameters. The material deformation tensor $\mathbf{F}$ is evaluated as $\mathbf{F} = \mathbf{I} + \nabla\mathbf{d}$ and its determinant is denoted by $J = \det(\mathbf{F})$.

2.2.2 Numerical Methods

An implicit trapezoidal rule time integration is used to solve the solid equations, where both the inertial and the surface forces are evaluated at the current time instant t^{n+1}:

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \frac{\Delta t}{2\rho_s}\left(\nabla\cdot\sigma_s^{n+1} + \nabla\cdot\sigma_s^n\right), \tag{10}$$

$$\mathbf{d}^{n+1} = \mathbf{d}^n + \frac{\Delta t}{2}\left(\mathbf{v}^{n+1} + \mathbf{v}^n\right). \tag{11}$$

where $\mathbf{v}$ approximates the first time derivative of the displacements $\mathbf{d}$, $\mathbf{v} = \frac{\partial \mathbf{d}}{\partial t}$.

The equations are discretized in space using a finite-volume method with a total Lagrangian approach. The momentum equation is integrated on the undeformed configuration. More details concerning the governing equations and numerical methods for this solver can be found in [4].

2.3 FSI coupling

2.3.1 Governing Equations

The coupling conditions on the fluid-structure interface are derived from the kinematic and dynamic equilibrium at the common boundary. For a no-slip type interface they read

$$\mathbf{u}_\Gamma = \frac{\partial \mathbf{d}_\Gamma}{\partial t}, \tag{12}$$

$$\sigma_s\mathbf{n}_\Gamma = \sigma_f\mathbf{n}_\Gamma \tag{13}$$

at Γ, where $\mathbf{n}_\Gamma$ is the unit normal vector on the interface.

2.3.2 Numerical Methods

A Dirichlet-Neumann domain decomposition approach is followed for the coupling of the fluid and the structure equations. Therefore, the fluid equations are solved for a known displacement of the interface

(Dirichlet boundary condition derived from kinematic equilibrium Eq. (12)), while the structure equations are solved for a known stress on the interface (Neumann boundary condition derived from dynamic equilibrium Eq. (13)).

A semi-implicit FSI coupling method is employed, as proposed in [10, 11], where only the fluid pressure term is strongly coupled to the structure via coupling iterations. The remaining fluid terms and the geometrical nonlinearities are only loosely coupled to the structure and, therefore, solved only once at every time step. We segregate the fluid pressure term by using a projection method to solve the fluid equations. It means that the Poisson equation (Eq. (5)) is solved several times per time step during the FSI coupling iterations, while the remaining fluid equations and the mesh moving are only executed once per time step. Moreover, as seen in section 2.1.2, Eq. (5) is the only part of the fluid discretized equations which is implicit in time. For this equation, we construct and solve a a linear system of equations. Thus, the repeated solution of the Poisson equation accounts for the majority of the computational cost of solving the fluid equations in the coupled FSI problem. Each time step of the coupled problem consists of performing mesh movement, convection and diffusion in the fluid solver and, subsequently, solving the fixed point equation

$$\begin{pmatrix} \mathcal{S}(\sigma_\Gamma) \\ \mathcal{P}(\mathbf{d}_\Gamma) \end{pmatrix} = \begin{pmatrix} \mathbf{d}_\Gamma \\ \sigma_\Gamma \end{pmatrix} \tag{14}$$

iteratively, where $\mathcal{S}$ is the mapping of the interface stress tensor σ_Γ to the interface displacement vector $\mathbf{d}_\Gamma$ by the structure solver and $\mathcal{P}$ the mapping of the interface displacement to the interface stress by solving the pressure Poisson equation and re-calculating stress in the fluid solver. To improve the stability of numerical solution and accelerate the convergence of the iterative solution of Eq. (14), we use a multi-vector quasi-Newton method, as proposed in [6, 12].

We exploit GPUs to accelerate the solution of the Poisson equation for the fluid (Eq. (5)). As a result of the semi-implicit method used in this work, this component constitutes a significant part of the total computational cost of solving the coupled FSI problem. The strategy introduced in the current work can be incorporated for other applications to improve the solution performance.

3 GPU Acceleration of Poisson Solver

In this section, we describe our novel setup for the efficient exploitation of hybrid machines for partitioned FSI simulations. As explained earlier, due to the semi-implicit coupling strategy, the Poisson solver in the fluid code is the most expensive component. SIMD operations are exploited within this solver. Therefore, the huge performance benefit by GPU-acceleration of this solver can be foreseen. The rest of computations in fluid solver is small compared to the Poisson solver. The solid solver is based on a linear de-coupling of the computations for x, y and z directions and uses an outer iteration for solution convergence. This implies that, the data chunks used by this solver for each iteration are small and the work done per iteration per data is not large enough to saturate GPU's compute capacity. Thus, offloading cost overcompensates the gain in computation speed due to GPU acceleration. Therefore, we carry out these computations on the CPUs.

The following steps are taken to efficiently exploit GPUs for FSI simulation:

1. Both solvers are initialized on the CPUs. Data structures are created, surface meshes are analyzed

for data mapping and establishing inter-solver communication channels via the coupling library.

2. The fluid solver solves the explicit part of the ALE equations and translate the mesh on the CPU, followed by constructing the pressure system of equations to be solved on GPUs. The solid solver constructs the linearized structural system of equations.
3. The structure system of equations is solved on the CPUs. Simultaneously, the linear systems given by Eq. (5) in the fluid solver are accelerated using GPUs. We use a PETSc KSP solver [13, 14] to iteratively solve these equations.
4. After each iteration, preCICE checks the convergence of the FSI coupling and data are communicated between the solvers. For this purpose, coupling interface data are exchanged between the single-physics solvers via TCP/IP communication between CPUs. This process is repeated until the FSI coupling convergence is achieved.
5. When the coupling is converged, the fluid solver calculates the velocity using the converged pressure field. This is the end of the time-step and the solver proceeds to the next time-step.

As explained earlier, we use a PETSc KSP solver to exploit GPUs for solving the fluid's Poisson equation. The PETSc's design separates the application code and the solver. This allows using a wide range of GPU programming models, such as Kokkos, CUDA and OpenCL. We employ a CUDA back-end since we run our simulations on NVIDIA GPUs in this work. In addition, the data are shared between the PETSc programming models and the application code and therefore no copy is needed. This improves the efficiency by saving the copy time [14].

4 Performance and Scalability Analysis

To show the performance of the coupled solver, we present strong scalability measurements for the simulation of blood flow inside a patient-specific aorta. In addition, we compare the simulation time with and without using GPUs for both the fluid solver and the overall coupled simulation. The essential parts of the current solver for the accuracy of the results are the same as those in [4]. Therefore, we do not provide any accuracy analysis in this work and focus only on the performance perspectives.

4.1 Test Case Description

The simulation of blood flow inside a patient-specific aorta is a challenging FSI problem both from a numerical and a computational point of view. Moreover, it is a very practical problem with direct application in biomedical engineering. In this test case, the geometry of the aorta is provided by the 2nd CFD challenge of the STACOM 2013 conference [15]. Figure 1 (left) shows the geometry of the aorta with inlet and outlet boundaries. The thickness of the aortic wall and its mechanical properties were not provided in [15], we assume $h = 2$mm, which is in the typical physiological range. For the mechanical properties of the wall, we use the density $\rho_s = 1200\text{kg}/\text{m}^3$, the Young modulus $E = 3 \times 10^5 \text{N}/\text{m}^2$, and the Poisson ratio $\nu = 0.3$. The density and the viscosity of the blood are assumed to be $\rho_f = 1000\text{kg}/\text{m}^3$ and $\mu_f = 0.004\text{Pa} \cdot \text{s}$, similar to the values used in [4, 16].

At the inlet, Dirichlet boundary conditions are used for the fluid velocity, using measured flow rate data provided in [15], combined with Neumann boundary conditions for the fluid pressure. At the outlet boundaries, explicit RCR Windkessel boundary conditions [17] are used to model the effect of the rest of the vascular network. The Windkessel parameters are chosen as those reported in [18]. For the structure,

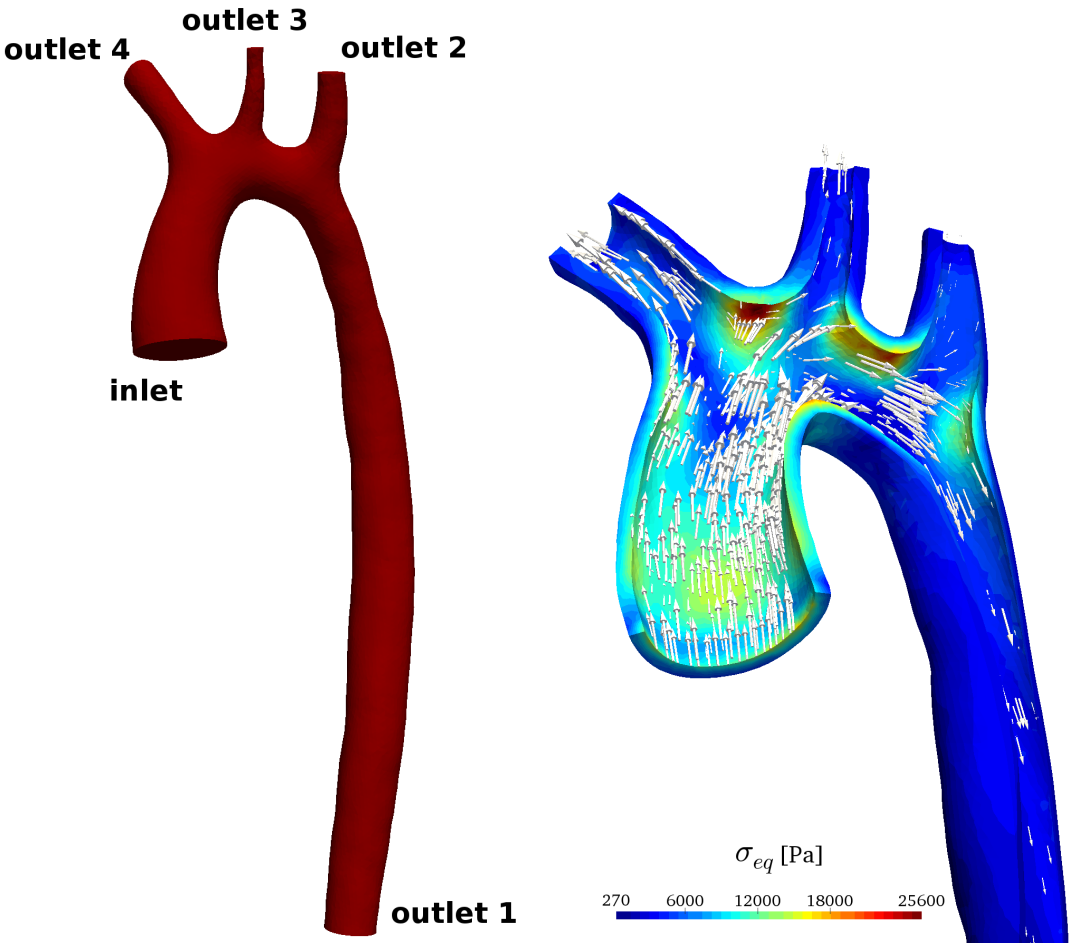

Figure 1: Aorta test case: left: 3D geometry of the patient-specific aorta (geometry provided in [15]); right: coupled FSI solution at $t = 0.06$, showing the fluid velocity vectors inside the deformed aortic wall and the von Mises equivalent stress at the wall.

a zero-displacement (clamped) boundary condition is used at the inlet and the outlets, while a traction-free boundary condition is assumed on the outer surface of the aortic wall. Unstructured tetrahedral meshes at two different resolutions (see Table 1) are used for the fluid and the solid domain. The fluid and the structure meshes match at interface. Figure 1 (right) shows the coupled FSI solution at a time instant $t = 0.06$s. The figure shows the velocity vectors inside the deformed aortic wall. The color contours in the structural domain correspond to the von Mises equivalent stress.

Table 1: Aorta test case: computational grids used for the scalability tests.

Mesh name	No. of cells		
	Fluid	Structure	Common Interface
M1	38M	20M	800K
M2	20M	9M	400K

4.2 Hardware Architecture and Numerical Library Description

The performance and scalability measurements are carried out on the Vulcan cluster at the High Performance Computing Center Stuttgart (HLRS). This cluster includes a machine which has 8 heterogeneous nodes. Each of these nodes consists of 2 $\times$ 2.6GHz Intel Xeon Gold 6240 (CascadeLake) with 36 cores in total and 8 $\times$ NVIDIA Tesla V100 SXM2 GPUs. The total available memory on each node is 768GB. We use the GNU GCC compiler and an Intel MPI implementation compatible with the GCC compiler for intra-solver parallelization and communication.

4.3 Numerical Performance Analysis

We initially present the performance analysis only for the fluid solver (within the coupled solver) to investigate the GPU speed-up individually on the accelerated solver. We run coupled FSI simulations using both meshes in Table 1 with and without GPU acceleration. Table 2 presents the fluid solver run–time and speed-up due to GPU acceleration for different GPU counts. We observe up to 133 times speed-up for the fluid solver in a coupled simulation. We see a higher speed-up for a smaller number of compute nodes as well as for the larger mesh. This is probably due to the larger data chunk sent to the GPUs which can better exploit the GPUs computation capacity.

Table 2: Hybrid CPU-GPU test case: fluid solver speed up by GPU acceleration (run-times measured in a coupled FSI simulation).

Mesh	number of CPUs-GPUs	computation time (s) with acceleration	computation time (s) without acceleration	speed-up
M1	8	15.9	2128.3	133.4
M1	16	12.1	1064.1	88.2
M2	8	8.3	992.3	119.0
M2	16	7.1	495.2	69.4

In addition, to investigate the GPU-acceleration effect on the overall performance of the coupled solver, we compare the coupled simulation run-times with and without GPU acceleration. Figure 2 depicts the

average run–time per coupling iteration for different computing node numbers for mesh M2. The number of nodes equals the total number of computing nodes exploited by the fluid and the solid solver together. In the hybrid simulation, the fluid solver exploits an equal number of CPUs and GPUs, i.e., from each node we exploit 8 GPUs and 8 CPU-cores. While, in a CPU-only scenario, it exploits all 36 available CPUs on each node. For solid solver, all of the 36 CPU-cores within each node are used, without any GPU acceleration. The node distribution is calculated using the load balancing method presented in [19]. For compute resources distribution, we always allocate a complete node to a solver to avoid node sharing between solvers. As seen in Figure 2, a very good reduction in computational time is achieved by increasing the number of compute nodes to 8, which is the full machine capacity. In addition, we observe 6 to 8 times faster simulations due to GPU acceleration of the fluid solver. In the current test-case, the fluid-solver's run–time is always smaller than the solid solver (in case of GPU-acceleration), which results in fluid solver's idling. We expect larger gain if more compute nodes were available, where we can do finer load balancing and avoid the mentioned idle time.

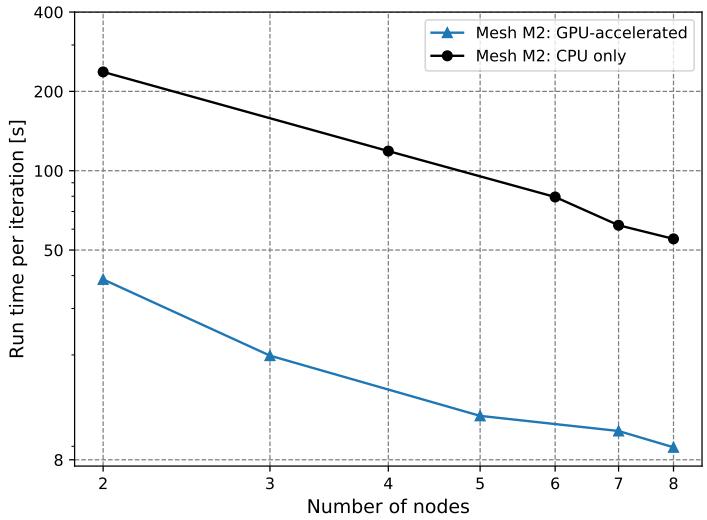

Figure 2: Aorta test case: average run–time per coupling iteration for different node counts with and without GPU-acceleration of the fluid solver (strong scalability test results using mesh M2).

5 Conclusion

We presented a semi-implicit partitioned fluid-structure interaction solver capable of efficiently exploiting hybrid CPU-GPU architectures. The partitioned approach allows using individual setups for different sub-solvers. Thus, we were able to port only the compute intensive kernels of our original CPU-only solver to GPUs. The offloading was completely transparent to the coupling library and did not interfere with other kernels, inter-code communication and coupling numerics. This preserved both the accuracy of the solution and the parallel efficiency of the other components.

A patient–specific aorta test-case is show-cased for performance analysis of the solver. GPU acceleration for various mesh sizes demonstrated an excellent run-time reduction. We observed up to 133 times speed-up in the fluid solver which resulted in 6 to 8 times faster total simulation compared to a CPU-only

scenario. In addition, a very good scalability was achieved for the coupled solver on up to 8 compute nodes (full machine).

Acknowledgment

We thank the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) for supporting this work by funding - EXC2075 – 390740016 under Germany's Excellence Strategy. We acknowledge the support by the Stuttgart Center for Simulation Science (SimTech).

This work was also financially supported by

- priority program 1648 - Software for Exascale Computing 214 (ExaFSA - Exascale Simulation of Fluid-Structure-Acoustics Interactions) of the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation),
- Ministerio de Economía y Competitividad, Secretaría de Estado de Investigación, Desarrollo e Innovación, Spain (ENE2017-88697-R).

The performance measurements were carried out on the Vulcan cluster at the High-Performance Computing Center Stuttgart (HLRS). The authors wish to thank HLRS for compute time and technical support.

REFERENCES

[1] J. Degroote, Partitioned simulation of fluid-structure interaction, Archives of Computational Methods in Engineering 20 (2013) 185–238. doi:10.1007/s11831-013-9085-5.

[2] G. Hou, J. Wang, A. Layton, Numerical methods for fluid-structure interaction - A review, Communications in Computational Physics 12 (2) (2012) 337–377. doi:10.4208/cicp.291210.290411s.

[3] S. Hewitt, L. Margetts, A. Revell, P. Pankaj, F. Levrero-Florencio, OpenFPCI: A parallel fluid–structure interaction framework, Computer Physics Communications 244 (2019) 469–482. doi:https://doi.org/10.1016/j.cpc.2019.05.016.

[4] A. Naseri, A. Totounferoush, I. González, M. Mehl, C. D. Pérez-Segarra, A scalable framework for the partitioned solution of fluid-structure interaction problems, Computational Mechanics 66 (2020) 471–489. doi:10.1007/s00466-020-01860-y.

[5] TermoFluids CFD software, www.termofluids.com (2020).

[6] H.-J. Bungartz, F. Lindner, B. Gatzhammer, M. Mehl, K. Scheufele, A. Shukaev, B. Uekermann, preCICE–a fully parallel library for multi-physics surface coupling, Computers & Fluids 141 (2016) 250–258. doi:10.1016/j.compfluid.2016.04.003.

[7] F. Jiang, K. Matsumura, J. Ohgi, X. Chen, A gpu-accelerated fluid–structure-interaction solver developed by coupling finite element and lattice boltzmann methods, Computer Physics Communications 259 (2021) 107661.

[8] R. Verstappen, A. Veldman, Symmetry-preserving discretization of turbulent flow, Journal of Computational Physics 187 (1) (2003) 343–368. doi:10.1016/S0021-9991(03)00126-8.

[9] F. X. Trias, O. Lehmkuhl, A. Oliva, C. D. Pérez-Segarra, R. Verstappen, Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids, Journal of Computational Physics 258 (2014) 246–267. doi:10.1016/j.jcp.2013.10.031.

[10] A. Naseri, O. Lehmkuhl, I. Gonzalez, E. Bartrons, C. D. Pérez-Segarra, A. Oliva, A semi-implicit coupling technique for fluid–structure interaction problems with strong added-mass effect, Journal of Fluids and Structures 80 (2018) 94–112. doi:10.1016/j.jfluidstructs.2018.03.012.

[11] A. Naseri, I. Gonzalez, A. Amani, C. D. Pérez-Segarra, A. Oliva, A second–order time accurate semi–implicit method for fluid–structure interaction problems, Journal of Fluids and Structures 86 (2019) 135–155. doi:10.1016/j.jfluidstructs.2019.02.007.

[12] K. Scheufele, M. Mehl, Robust multisecant quasi-Newton variants for parallel fluid-structure simulations–and other multiphysics applications, SIAM Journal on Scientific Computing 39 (5) (2017) S404–S433. doi:10.1137/16M1082020.

[13] V. Minden, B. Smith, M. G. Knepley, Preliminary implementation of petsc using gpus, in: GPU solutions to multi-scale problems in science and engineering, Springer, 2013, pp. 131–140.

[14] R. T. Mills, M. F. Adams, S. Balay, J. Brown, A. Dener, M. Knepley, S. E. Kruger, H. Morgan, T. Munson, K. Rupp, et al., Toward performance-portable petsc for gpu-based exascale systems, arXiv preprint arXiv:2011.00715 (2020).

[15] 2nd CFD challenge predicting patient-specific hemodynamics at rest and stress through an aortic coarctation, http://www.vascularmodel.org/miccai2013/ (2013).

[16] M. A. Fernández, M. Landajuela, M. Vidrascu, Fully decoupled time-marching schemes for incompressible fluid/thin-walled structure interaction, Journal of Computational Physics 297 (2015) 156–181. doi:10.1016/j.jcp.2015.05.009.

[17] N. Westerhof, J.-W. Lankhaar, B. E. Westerhof, The arterial Windkessel, Medical & biological engineering & computing 47 (2) (2009) 131–141. doi:10.1007/s11517-008-0359-2.

[18] S. Pant, B. Fabrèges, J.-F. Gerbeau, I. Vignon-Clementel, A methodological paradigm for patient-specific multi-scale cfd simulations: from clinical measurements to parameter estimates for individual analysis, International journal for numerical methods in biomedical engineering 30 (12) (2014) 1614–1648. doi:10.1002/cnm.2692.

[19] A. Totounferoush, N. Ebrahimi Pour, J. Schroder, S. Roller, M. Mehl, A new load balancing approach for coupled multi-physics simulations, in: In Proceedings of IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Rio de Janeiro, Brazil, May 2019. doi:10.1109/IPDPSW.2019.00115.