Fluid-Acoustics Interaction on Massively Parallel Systems

Hans-Joachim Bungartz, Harald Klimach, Verena Krupp, Florian Lindner, Miriam Mehl, Sabine Roller, and Benjamin Uekermann

Abstract To simulate fluid-acoustics interaction, we couple inviscid Euler equations in the near-field, which is relevant for noise generation, to linearized Euler equations in the far-field. This allows us to separate the critical scales and treat each domain with an individual discretization. Both fields are computed by the high-order discontinuous Galerkin solver Ateles, while we couple the solvers at the interface by the library preCICE. We discuss a detailed performance analysis of the coupled simulation on massively parallel systems. Furthermore, to show the full potential of our approach, we simulate a flow around a sphere.

1 Introduction

Simulation of fluid-structure-acoustics interaction (FSA) will bring new insight into different applications, as, for example, the sound design of aircrafts or wind energy plants. Fluid-acoustics interaction, an important milestone towards full FSA, yields a multi-scale problem including different lengths and timescales, which is numerically challenging as computation on the finest resolution is not feasible. However, different phenomena typically appear in spatially separated domains. Thus, it is possible to decompose the overall simulation domain into non-overlapping partitions with distinctive treatment and link these partitions via surface coupling. We aim to develop a partitioned simulation, reusing existing software for each individual discipline. This allows the usage of different numerical methods and tailored grid resolutions for each partition. Moreover, we can benefit from prior

H.-J. Bungartz • B. Uekermann (✉)
Technische Universität München, Boltzmannstraße 3, Garching b. München, Germany
e-mail: bungartz@in.tum.de; uekerman@in.tum.de

H. Klimach • V. Krupp • S. Roller
Universität Siegen, Hölderlinstraße 3, Siegen, Germany
e-mail: harald.klimach@uni-siegen.de; verena.krupp@uni-siegen.de; sabine.roller@uni-seigen.de

M. Mehl • F. Lindner
Universität Stuttgart, Universitätsstraße 38, Stuttgart, Germany
e-mail: miriam.mehl@ipvs.uni-stuttgart.de; florian.lindner@ipvs.uni-stuttgart.de

M. Mehl et al. (eds.), *Recent Trends in Computational Engineering - CE2014*,
Lecture Notes in Computational Science and Engineering 105,
DOI 10.1007/978-3-319-22997-3_9

experience on how to scale up each single simulation on parallel systems. Coupling between physical solvers, however, needs to be carried out carefully to get a stable overall simulation, while not degrading the scalability.

Contrary to our approach, classical fluid-acoustics interaction simulation use a volume coupling, based on Lighthill's analogy [7]. Typically, a uni-directional coupling is applied, where the flow solver computes acoustic source terms that are fed to the afterwards separately executed acoustic wave propagation. Similar ideas are used in [6]. Such a volume coupling, however, hinders the separation of scales. For large scenarios where a detailed resolution is only necessary in a subdomain, a surface-coupled approach is advantageous. In our approach, we aim for a direct numerical simulation since each domain can be tailored to the physics and the usage of a two-way coupling allows full interaction. Similar direct numerical simulations without a coupling tool are, for example, done in [10].

The usage of today's supercomputers and tomorrow's exa-scale systems is indispensable, as only the careful resolution of all scales fully reveals the appearing phenomena. Applications such as wind energy plants, where only the noise disturbance in some large distances is of interest, easily reach the capacity of today's supercomputers. To make efficient use of the computational resources, all components of a partitioned simulation must scale. We couple the high order discontinuous Galerkin solver Ateles, included in the APES framework [8], by means of the coupling library preCICE [3]. Ateles proved scalability up to complete supercomputers [11], while preCICE uses a pure point-to-point approach, not depending on any central coupling instance, which could deteriorate the solver scalability.

In this work, we study the performance of a simulation of such a coupled fluid-acoustics interaction and draw conclusions on the applicability of our approach to challenging engineering applications. We describe, therefore, the physics and numerics of our approach in Sect. 2, and the used software in Sect. 3. Afterwards, we first study an academic performance test case in Sect. 4, and show then, in Sect. 5, the full simulation of a flow around a sphere where acoustic waves are generated around the geometry and propagated into the far field.

2 Physics and Numerics

In this section, we first describe the physical models of our setting and then the numerical methods that we apply.

2.1 Physics

The governing equations for flow phenomena are described by the conservation laws of mass, momentum and energy. The complete conservation laws are stated

in the compressible Navier-Stokes equations. A simplification can be obtained by neglecting the dissipation terms. This leads to the inviscid compressible Euler equations and can be treated in the conservative form

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F_x}(\mathbf{U})}{\partial x} + \frac{\partial \mathbf{F_y}(\mathbf{U})}{\partial y} + \frac{\partial \mathbf{F_z}(\mathbf{U})}{\partial z} = 0\,, \tag{1}$$

with the state vector $\mathbf{U}$ and the flux functions $\mathbf{F_x}(\mathbf{U})$, $\mathbf{F_y}(\mathbf{U})$ and $\mathbf{F_z}(\mathbf{U})$, defined as follows:

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ \rho v_z \\ e \end{pmatrix}, \mathbf{F_x}(\mathbf{U}) = \begin{pmatrix} \rho v_x \\ \rho v_x^2 + p \\ \rho v_x v_y \\ \rho v_x v_z \\ v_x(e+p) \end{pmatrix}, \mathbf{F_y}(\mathbf{U}) = \begin{pmatrix} \rho v_y \\ \rho v_y v_x \\ \rho v_y^2 + p \\ \rho v_y v_z \\ v_y(e+p) \end{pmatrix}, \mathbf{F_z}(\mathbf{U}) = \begin{pmatrix} \rho v_z \\ \rho v_z v_x \\ \rho v_z v_y \\ \rho v_z^2 + p \\ v_z(e+p) \end{pmatrix}. \tag{2}$$

ρ denotes the density, $\mathbf{v}$ the velocity vector, p the pressure and e the energy. To fully describe the system, the assumption of an ideal gas yields the relation between p and e:

$$p = \rho R T = (\gamma - 1)\left(e - \frac{\rho \mathbf{v} \cdot \mathbf{v}}{2}\right),$$

where γ is the adiabatic exponent, T the temperature, and R the specific gas constant.

If there are only small changes in the flow, the flow variables can be separated into mean and perturbation. The perturbation describes the acoustic phenomena and is denoted with a prime in the state vector

$$\mathbf{U}' = \begin{pmatrix} \rho' \\ v_x' \\ v_y' \\ v_z' \\ p' \end{pmatrix}.$$

The linearized Euler equations are now defined similar to (1), and the flux functions are directly evaluated around the mean flow $\left(\rho\ v_x\ v_y\ v_z\ p\right)^T$:

$$\mathbf{F_x}(\mathbf{U}') = \begin{pmatrix} v_x\rho' + \rho v_x' \\ v_x v_x' + \frac{1}{\rho}p' \\ v_x v_y' \\ v_x v_z' \\ v_x p' + \gamma p v_x' \end{pmatrix}, \mathbf{F_y}(\mathbf{U}') = \begin{pmatrix} v_y\rho' + \rho v_y' \\ v_y v_x' \\ v_y v_y' + \frac{1}{\rho}p' \\ v_y v_z' \\ v_y p' + \gamma p v_y' \end{pmatrix}, \mathbf{F_z}(\mathbf{U}') = \begin{pmatrix} v_z\rho' + \rho v_z' \\ v_z v_x' \\ v_z v_y' \\ v_z v_z' + \frac{1}{\rho}p' \\ v_z p' + \gamma p v_z' \end{pmatrix}. \tag{3}$$

Typically, there is a domain around a complex geometry where the flow is turbulent and sound is generated. For this regime, fine meshes are needed and the full set of non-linear equations (1) has to be solved. Further away from the geometry, the propagation of the acoustic waves prevails. Here, the mesh can be coarser, and the set of equations can be reduced to the linearized Euler equations (3). At the fluid-acoustics interface, the state variables can be transferred via Dirichlet boundary conditions to the flow respectively the acoustic partition. The complete state is exchanged, whereas for the acoustic partition, we subtract the mean flow and only solve for the perturbations, the acoustic phenomena.

2.2 Numerics

2.2.1 Discontinuous Galerkin

We use a strong stability-preserving second order Runge-Kutta method to discretize in time [4]. In space, we use the discontinuous Galerkin (DG) method. Marking a combination of a finite element and a finite volume scheme, the discretization is based on a polynomial representation within an element and flux calculation between elements. The computational domain Ω is subdivided into non-overlapping grid-cells Q_i. The variational formulation reads

$$\partial t \int_{\Omega_i^n} \mathbf{U} \cdot \varphi d\mathbf{x} dt - \int_{\Omega_i^n} \mathbf{F} \nabla \varphi d\mathbf{x} dt + \int_{\partial \Omega_i^n} \mathbf{F}_n \cdot \varphi ds dt = 0 \,, \tag{4}$$

where $\partial \Omega_i^n$ denotes the boundary of a grid-cell, $\mathbf{F}_n$ the numerical flux over the boundary and φ the test function known from the classical finite element method. As basis functions, we choose Legendre polynomials. Due to their orthogonality and recursive definition, they lead to a fast evaluation and sparse structure of the mass and stiffness matrices. Furthermore, we use a modal approach due to computational advantages: The numerical flux can be directly evaluated in modal space and, using cubical elements, no extra transformation to the reference element is required. Using a high order scheme, now, results in several advantages: First, we get low numerical dissipation and a low dispersion error, which is particularly well-suited for computing the acoustic far field, where waves travel over long distances. Second, we get faster error convergence for smooth solutions. Last, a high order scheme results in less degrees of freedom maintaining the same accuracy. Hence, with regard to partitioned coupling, where data at the interface needs to be exchanged and causes communication, a reduced amount of data is beneficial.

2.2.2 Coupling Scheme

To discretize the Dirichlet boundary conditions at the coupling interface in time, we use a parallel explicit coupling, combined with a fixed timestep size for both

domains. This means that both solvers can be executed in parallel and exchange data at the coupling interface after each timestep. The complete explicit scheme of Ateles and the constant timestep size in both domains allows for a perfect a priori load-balancing, which can be adapted by a simple try-and-error over the first couple of timesteps. As a trade-off, we have to fix the timestep size in the Euler domain, which can result in a loss of efficiency for a varying CFL condition. At the coupling interface we use matching meshes. A nearest-neighbor mapping guarantees that the matching vertices, which can be decomposed differently on both solvers, are assigned correctly.

3 Software

In this section, we have a closer look at the used software, the discontinuous Galerkin solver Ateles and the coupling library preCICE.

3.1 Ateles

Ateles is an explicit-in-time discontinuous Galerkin solver that is specifically geared towards high order schemes. It is part of the APES suite [8], which provides tools for pre- and post-processing on the basis of the common mesh library TreElM.[1] The TreElM library [5] relies on an octree representation of the mesh and provides the distributed neighborhood search within that mesh. The meshing tool provides arbitrary mesh configurations including multiple refinement levels. Since the meshes only consist of cubical elements, Ateles provides an embedded high order representation of material properties, which are used to represent complex geometries. Ateles supports the solution of various equation systems, including the ones used in this contribution, the inviscid Euler equations (1) and their linearization (3).

The APES framework is designed to take advantage of massively parallel systems. The domain decomposition of the octree mesh gives hierarchically structured data and using a space-filling curve maintains locality. This locality can be perfectly exploited by the DG scheme, which is strongly coupled for data within one element but only loosely at element boundaries. Moreover, the two levels of operation offered by the DG scheme can be exploited by hybrid parallelization. Data parallelism within elements with a tight coupling of degrees of freedom can be exploited by OpenMP threads, while partitions of elements can be computed in parallel by MPI processes. By free choices of the spatial scheme order and the hybrid parallelism, the solver can be adapted to the executing machine. Ateles scales up to

[1] https://bitbucket.org/apesteam/treelm.

32,768 processes without drastic loss of parallel efficiency on the IBM Dataplex machine SuperMUC at the Leibniz Computing Centre (LRZ) in Munich. Good scalability is also shown on other supercomputers such as the BG/Q System Juqueen at the Jülich Supercomputing Centre (JSC) and the Hornet at the High Performance Computing Centre Stuttgart (HLRS).

3.2 preCICE

preCICE[2] is an open source library for flexible surface-coupling of single-physics solvers, developed at the Technische Universität München and the Universität Stuttgart. It uses a black-box coupling approach, only requiring input-output information from single-physics solvers, and, thus, enables the coupling of commercial closed-source software as well. Code integration is minimal invasive and can be realized in only 30 lines of code. preCICE offers a high-level API, meaning that various coupling schemes, including parallel and serial schemes, explicit and implicit schemes, multi-coupling schemes, and subcycling are configurable at runtime, i.e., need no changes to the solver adapter. Figure 1 gives a brief overview on

```
1  SolverInterface precice("SolverName",rank,threadSize);
2  precice.configure("configuration.xml");
3   ...setup solver data structures
4  int wDataID = precice.getDataID("WriteDataName");
5  int rDataID = precice.getDataID("ReadDataName");
6  int[] vertIDs = precice.setMeshVertices(meshID,size,coords);
7  double preciceMaxDt = precice.initialize()
8  while (precice.isOngoing()){
9     dt = min(preciceMaxDt, solverDt);
10     ...compute solver time step using dt, compute wData
11    precice.writeBlockVectorData(wDataID,size,vertIDs,wData);
12    preciceMaxDt = precice.advance(dt);
13    precice.readBlockVectorData(rDataID,size,vertIDs,rData);
14     ...set rData
15 }
16 precice.finalize();
17  ...tear down solver data structures
```

Fig. 1 Example for adapting a solver to preCICE. All coupling numerics, data mapping schemes and the data exchange happens within the method "advance" in line 12. The original single-physics code is marked in *grey*

[2] http://www5.in.tum.de/wiki/index.php/PreCICE_Webpage.

how the main preCICE API functions can be used to couple a single-physics solver. Section 4 gives more details and a performance analysis about the main steering functions "initialize" and "advance".

preCICE offers various data mapping schemes for non-matching interface meshes, ranging from projection-based schemes to radial-basis function interpolation. Coupling schemes include sophisticated quasi-Newton schemes for strongly-coupled fluid-structure interaction as well as simple explicit schemes. Furthermore, advanced techniques, such as a restart mechanism, a Python callback interface, and an automatic generation of the configuration reference, are supported. preCICE has been coupled successfully to, amongst others, the commercial tools ANSYS Fluent and COMSOL [3], the in-house code ALYA [9], and the open-source code OpenFOAM [2]. preCICE is written in C++ and features a clean and modern software design with extensive unit and integration tests while maintaining minimal external dependencies and easy extensibility. Besides C++, interfaces in Fortran90/95 and Fortran2003 are offered. For more information, the reader is referred to [3].

In recent development, the parallel approach of preCICE has been ported to a pure point-to-point approach to allow for massively parallel simulations. Figure 2 depicts a schematic drawing of the parallel concept. The three feature groups, equation coupling, communication, and data mapping, sketched in the middle of Fig. 2, are executed on distributed data. Simple equation coupling schemes and projection-based mapping schemes have already been ported to this new outline, whereas quasi-Newton coupling schemes as well as radial-basis-function interpolation are subject of recent work. Parallel communication is still executed in a gather-scatter manner, meaning that all communication runs through the master process of each solver. While we are currently working on a local point-to-point

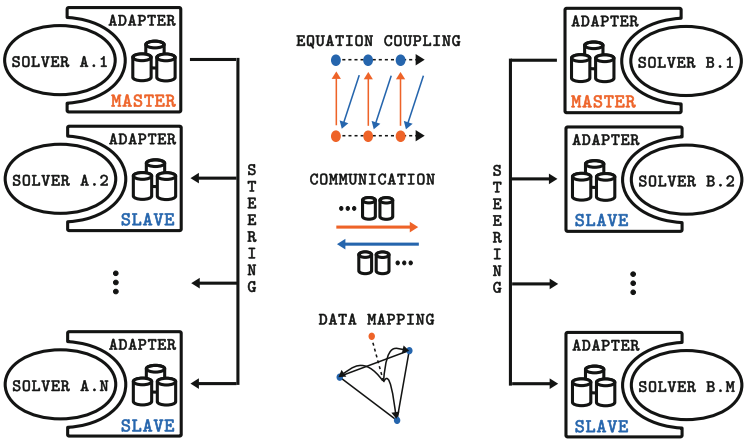

Fig. 2 Schematic drawing of the parallel concept of preCICE. Data is stored directly at the solvers processes. Thus, reading and writing data does not require communication. The three feature groups, depicted in the middle, are executed on distributed data

communication scheme (based on [1]), Sect. 4 gives detailed information on how far we can already reach with the gather-scatter approach.

4 Performance Test Case

In this section, we present a brief performance study for a coupled simulation.

4.1 Test Case Description

A cube with side length 8.0 m is cut into two equal-sized halves, orthogonal to the x-axis. In both domains, the flow solver Ateles solves for the Euler equations, while the coupling library preCICE is used to couple at the common, artificial interface, compare Fig. 3. A Gaussian density pulse travels with a constant speed from one domain into the other. Figure 4 shows the smooth transition of the pulse through the coupling interface. We perform a strong scaling with a fixed mesh-level $l = 5$, and a fixed polynomial degree $p = 32$ yielding an approximation order $O(33)$. This results in a total of $(p+1)^3 \cdot 2^{3(l-1)} \approx 1.5 \cdot 10^8$ degrees of freedom, including $(p+1)^2 \cdot 2^{2(l-1)} \approx 2.7 \cdot 10^5$ located at the coupling interface (per side). Since we know that Ateles itself shows a very good scalability per timestep for such a setting [11], we want to study how much the coupling deteriorate the performance. Furthermore, we want to get more insight on how long the initialization of preCICE takes, to get a rough estimate for production runs, like the one presented in Sect. 5.

All tests were computed on the thin-nodes partition of SuperMUC,[3] holding Intel Sandy Bridge-EP Xeon E5-2680 8C processors with 32 GB RAM. A node com-

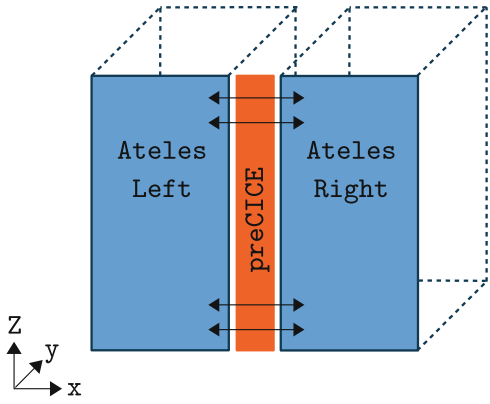

Fig. 3 Scenario set-up for the performance test case. A cube is cut into two equal-sized halves. Two Ateles instances are coupled via preCICE at the artificial coupling interface

[3] http://www.lrz.de/services/compute/supermuc/.

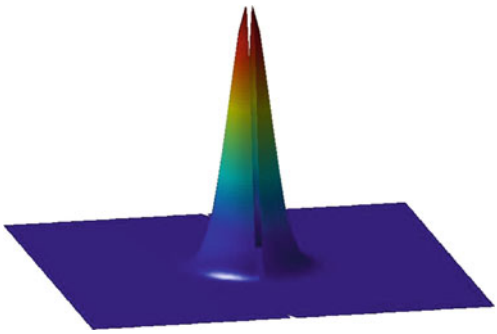

Fig. 4 Smooth transition of the density pulse through the coupling interface

prises 16 physical processors, and multiple nodes are connected by an Infiniband FDR10. For this machine, we can compare scalability results to pure Ateles runs from the Extreme Scaling Workshop, one major reason, why we chose SuperMUC for this work: A weak scaling of Ateles, using a single node with 16 MPI processes as reference and 65,536 elements per process, shows a drop in performance by a few percent when leaving the single node. After this initial drop when making use of the network, the performance stays fairly constant even up to 32,768 processes. A strong scaling using 262,144 elements, shows that the performance achieved per node with 32,768 processes on 4 islands is again comparable to the one on 4 nodes and close to the one on a single node without communication over the network.

Below, we present the performance results for the coupled simulation in two steps: first, we look at the initialization, and then, second, at the time per timestep—in preCICE notion "advance".

4.2 Initialization

In the following, the major steps of the two preCICE initialization function calls are listed, namely "initialize" and "initializeData" (compare Fig. 5). M_R and M_L refer to the master processes of Ateles Left and Ateles Right, whereas S_L and S_R denote individual slaves. Furthermore, Γ_L and Γ_R, are the left and right surface mesh.

initialize The communication between both participants is established. M_L gathers Γ_L of all S_L. Γ_L is then communicated from M_L to M_R. In the meanwhile, each S_R computes a bounding box of its local Γ_R and sends this information to M_R. M_R uses this information to pre-filter the global Γ_L individually for each S_R and send the reduced mesh to every S_R, accordingly. Each S_R, then, computes preliminary mappings between its pre-filtered Γ_L and its local Γ_R. Afterwards, the pre-filtered Γ_L is filtered a second time, such that only those vertices are left that have an influence on one of the mappings. Each S_R, now, possesses a local part of each mesh, Γ_L and Γ_R. The concrete information, which vertex is present on which S_R is sent back to M_R. The preliminary mapping is discarded.

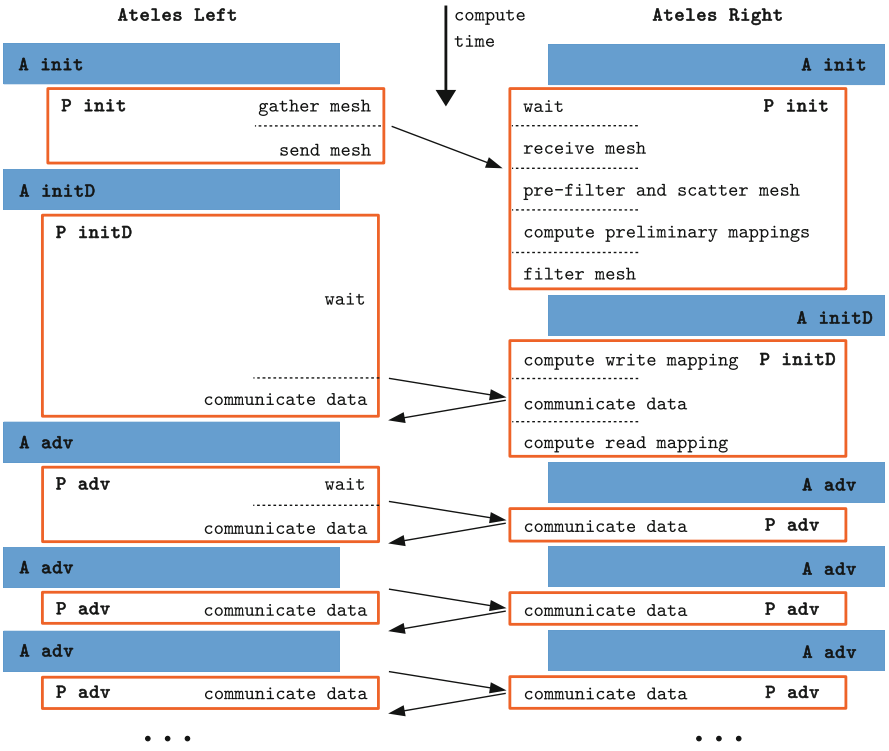

Fig. 5 Sketch of the major computational steps of the coupled simulation. Ateles computations are marked in *blue*, preCICE in *orange*. For sake of clarity we group the Ateles computation similar to preCICE into the sections "initialize", "initializeData", and "advance"

initializeData Each S_R computes the local write mapping. Afterwards, data on Γ_L is exchanged, from right to left, then from left to right. Finally, each S_R computes the read mapping.

Figure 6 shows an overview of the runtimes during initialization. In Table 1, the concrete numbers are listed.[4] The runtime of "initialize" comprises of "gather mesh", "communicate mesh", "scatter mesh", and "filter mesh", and shows no significant overhead for a rising number of processors. "initialize data", dominated by two mapping computations, scales nearly quadratically with the number of processors. To fully understand those two results, we discuss each step individually in the rest of this subsection.

gather mesh M_L gathers Γ_L. Therefore, each S_L sends its partition of Γ_L to M_L. This N:1 communication results in a bottleneck at M_L. The runtime is, however,

[4]Runtimes are averaged over 4 runs whereas the longest run is dropped.

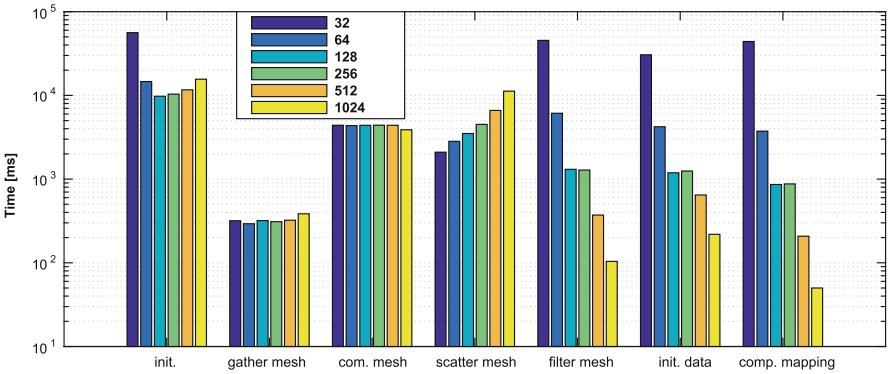

Fig. 6 Strong scaling for the major initialization steps. "initialize", comprising "gather mesh", "communicate mesh", "scatter mesh", and "filter mesh", shows no significant overhead for an increasing parallelization, whereas "initializeData", dominated by two mapping computations scales nearly quadratically. The number of processors listed corresponds to a single Ateles instance

Table 1 Explicit runtimes for Fig. 6

Total processors	**32**	**64**	**128**	**256**	**512**	**1024**
Processors at interface	16	32	64	64	128	256
Vertices per processor	17,424	8712	4356	4356	2178	1089
Initialize [ms]	**56,189**	**14,643**	**9766**	**10,348**	**11,653**	**15,613**
Gather mesh [ms]	318	293	319	310	323	385
Communicate mesh [ms]	4399	4361	4390	4406	4392	3884
Scatter mesh [ms]	2097	2826	3504	4505	6613	11240
Filter mesh [ms]	45,408	6119	1309	1282	372	104
Initialize data [ms]	**30,561**	**4222**	**1190**	**1248**	**646**	**219**
Compute mapping [ms]	44,058	3740	862	876	208	50

In the top rows, the number of processors at the coupling interface per total number of processors (per Ateles instance) and the number of vertices per interface processor are listed. Please note, that the number of interface processors stays constant when the total amount of processors is raised from 128 to 256

dominated by the total amount of data such that the rising granularity plays no role yet. The runtime stays nearly constant.

communicate mesh M_L sends the complete Γ_L to M_R. As this step is completely independent from the number of processors, we can observe the expected constant runtime. Slight deviations are due to waiting times as those numbers are measured at Ateles Right and the initialization load is not perfectly balanced. The overall time is significantly larger than the time spent in "gather mesh" because M_R creates the data structures of Γ_L on the fly. A clear separation of receiving and creating could safe some more initialization time at this point.

scatter mesh M_R pre-filters and sends Γ_L to each S_R individually, according to a bounding-box defined by each S_R. As this step leads to a compute load solely for M_R, we can observe some significant overhead, which tends to rise linearly with

the number of processors. Till 1024, however, this effort still lies around 10 s, and is, therefore, tolerable.

filter mesh Each S_R computes preliminary mappings between the local Γ_L and Γ_R and filters Γ_L accordingly. As no synchronization is needed, this steps is embarrassingly parallel. The applied nearest neighbor mapping has a computational cost of $O(N \cdot M)$ where N and M are the number of vertices of Γ_L and Γ_R. Thus, we can observe a nearly quadratic scaling. Please note that the number of processors at the coupling interface stays constant at 64 if the total number of overall processors per Ateles instance is raised from 128 to 256. The significantly higher costs for 32 processors is supposably due to memory hierarchy.

compute mapping Analogue to the last item, we can observe a nearly quadratic scaling.

4.3 Advance

After each timestep, preCICE maps data between both meshes and exchanges data between both participants. The mapping is embarrassingly parallel, whereas the data exchange is performed in a gather-scatter-manner. Thus, we do not expect the pure preCICE runtime to scale. In this subsection, we discuss, at which point the preCICE overhead becomes significant for the overall simulation. Figure 7 (left) shows the compute time per timestep split into Ateles and preCICE. The time spent in preCICE is not significant up to about 1024 processors per Ateles instance. Figure 7 (right) shows the parallel efficiency that we get up to 1024 processors, and extrapolated values afterwards. We still expect around 80 % parallel efficiency for 2000 processors per Ateles instance. With our recent work on a true point-to-point

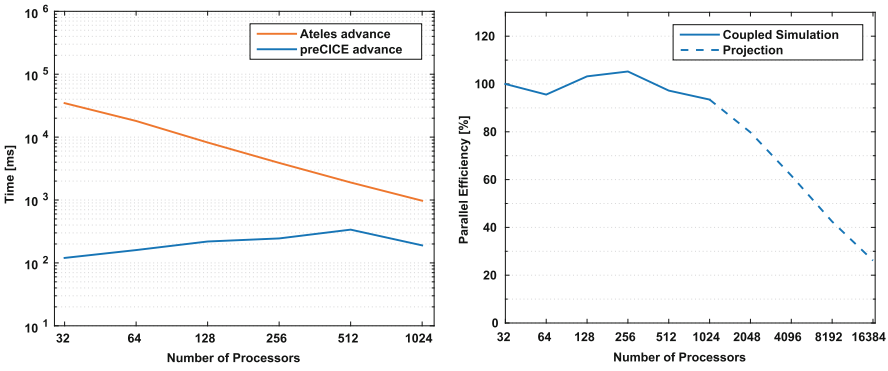

Fig. 7 *Left*: comparison of the time spent in Ateles and in preCICE during one timestep. *Right*: parallel efficiency of the overall simulation during one timestep, measured values up to 1024 processors (per Ateles instance) and extrapolation based on those values afterwards (perfect Ateles scaling and constant preCICE "advance" assumed)

communication scheme (cf. [1]), we are confident to get rid of this bottleneck as well.

4.4 Conclusions

In this section, we analyzed the performance of a simple, academic test case to get more insight into the scaling of a coupled simulation. Still, the testcase marks a challenging example, as it holds a high number of degrees of freedom, $1.5 \cdot 10^8$, and a significant coupling interface, a cut through the complete domain. Nevertheless, we are able to keep the initialization effort of the coupling below 15 s, which is acceptable for most real-world engineering applications. At runtime, the time spent per timestep is dominated by the computations of Ateles, meaning that the coupling does not deteriorate the overall scalability for up to 1024 processors per participant, though preCICE uses a gather-scatter data transfer. Both parts, initialization and time per timestep, leave room for improvement, but we conclude this section by stating that preCICE can successfully deal with coupled engineering applications on a high, but not yet very high level of parallelism.

5 Application Scenario: Flow Around a Sphere

To test the full potential of our approach, we run a typical production example, a classical flow around a sphere. The Euler domain has a length of 2 m, along with a width and height of 0.9 m, and is surrounded by the acoustic domain, which is 1.9 m in y and z direction. Inside the Euler domain, at $x = 0.75$ m a sphere with diameter 0.3 m is located. The density is set to $\rho = 1.4$ kg/m^3, and the pressure to $p = 1.0$ N/ m^2. A constant flow in x direction of $v_x = 0.1$ m/s drives the flow. The same values are set as mean flow properties for the acoustic domain. The flow around the sphere generates acoustic waves, which travel in each direction. The sphere itself is realized via embedded material properties.

The Euler and acoustic domains are discretized with 1620 and 5600 elements respectively, while we count a total of 720 elements at the interface. Both domains use a $O(10)$ DG scheme. The timestep size is set to 10^{-5} s. From simple trial-and-error tests over a few timesteps, we observed that the full Euler equations are five times more expensive to solve per element than the linearized equations. This shows the benefit of a partitioned approach. We use, therefore, 810 processors in the Euler domain (two elements per processor), and 560 processors in the Acoustic domain (ten elements per processor), to achieve a decent load-balancing. This results in 360 processor at the interface in the Euler domain. Figure 8 shows pressure values at $t = 0.4$ s on a cut through the domain.

A simulation time of 0.4 s results in a compute time of 8.2 h. For the first five timesteps, we measure 870 ms per timestep for the pure physical computations

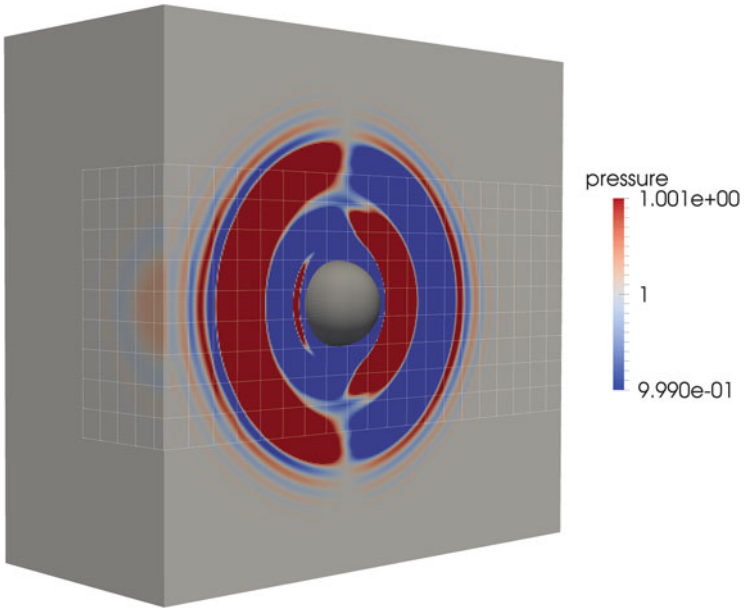

Fig. 8 Coupled simulation of a flow around a sphere. The plot visualizes pressure values at $t = 0.4$ s on a cut through the domain. To distinguish both domains, we show the mesh wireframe in the Euler domain

in each domain, while the average preCICE "advance" call takes 558 ms. This higher ratio of coupling effort to computation, compared to Sect. 4, is mainly due to the high ratio of processors at the interface in the Euler domain and the lower polynomial order.

6 Summary and Outlook

We coupled near-field Euler equations to linearized Euler equations in the far-field. This separation allows us to use a tailored discretization length and order for each individual field. Both fields are simulated by the high-order discontinuous Galerkin solver Ateles, while we use the coupling library preCICE to couple the solvers at the common coupling interface. Recently, we changed the parallel approach of preCICE to a pure point-to-point scheme, not using any central instance. In this work, we first studied the performance of our coupled approach by means of a simple academic test problem, a travelling density pulse. We used up to 1024 processors per Ateles instance and did not observe any significant coupling overhead at initialization. Though we count a total amount of $1.5 \cdot 10^8$ degrees of freedom, we are able to establish the coupling in 15 s. The time per timestep, on the other hand, is dominated by Ateles, thus, the coupling does not deteriorate the overall scalability and we

can still observe over 90 % parallel efficiency. We work recently on an improved communication scheme in preCICE, which should allow us to reach beyond 1000 processors.

Afterwards, to show the full potential of our approach, we simulated the flow around a sphere. The costs in the Euler domain were five times higher than in the acoustic domain, showing the benefit of the partitioned approach. Due to the challenging structure of the interface—every second processor in the Euler domain is located at the interface—we observed a relatively higher coupling overhead, compared to the performance test case. The coupling costs were, however, still lower than the computational costs, which keeps them tolerable. We plan to study further application scenarios in future work.

Acknowledgement The financial support of the Institute for Advanced Study (IAS) of the Technische Universität München, and of SPPEXA, the German Science Foundation Priority Programme 1648—Software for Exascale Computing is thankfully acknowledged.

References

1. Atanasov, A.: Software idioms for component-based and topology-aware simulation assembly and data exchange in high performance computing and visualisation environments. Ph.D. thesis, Technische Universität München, Institut für Informatik (2015)
2. Blom, D.S., Krupp, V., van Zuijlen, A.H., Klimach, H., Roller, S., Bijl, H.: On parallel scalability aspects of strongly coupled partitioned fluid-structure-acoustic interaction. In: Proceedings of 6th International Conference on Computational Methods for Coupled Problems in Science and Engineering, pp. 1–10, Venice (2015)
3. Gatzhammer, B.: Efficient and flexible partitioned simulation of fluid-structure interactions. Ph.D. thesis, Technische Universität München (2015)
4. Gottlieb, S., Shu, C.W., Tadmor, E.: Strong stability-preserving high-order time discretization. SIAM Rev. **43**(1), 89–112 (2001)
5. Klimach, H.G., Hasert, M., Zudrop, J., Roller, S.P.: Distributed octree mesh infrastructure for flow simulations. In: Eberhardsteiner, J. (ed.) ECCOMAS 2012 - European Congress on Computational Methods in Applied Sciences and Engineering. Vienna (2012)
6. Kornhaas, M., Schäfer, M., Sternel, D.C.: Efficient numerical simulation of aeroacoustics for low mach number flows interacting with structures. Comput. Mech. **55**(6), 1143–1154 (2015)
7. Lighthill, M.: On sound generated aerodynamically. I. General theorie. Proc. R. Soc. Lond. A **211**, 564–587 (1952)
8. Roller, S., Bernsdorf, J., Klimach, H., Hasert, M., Harlacher, D., Cakircali, M., Zimny, S., Masilamani, K., Didinger, L., Zudrop, J.: An adaptable simulation framework based on a linearized octree. In: Resch, M., Wang, X., Bez, W., Focht, E., Kobayashi, H., Roller, S. (eds.) High Performance Computing on Vector Systems 2011, pp. 93–105. Springer, Berlin/Heidelberg (2012)
9. Uekermann, B., Cajas, J.C., Gatzhammer, B., Houzeaux, G., Mehl, M., Vázquez, M.: Towards partitioned fluid-structure interaction on massively parallel systems. In: Proceedings of WCCM XI/ ECCM V/ ECFD VI, Barcelona (2014)
10. Utzmann, J.: A domain decomposition method for the efficient direct simulation of aeroacoustic problems. Ph.D. thesis, Universtität Stuttgart (2008)
11. Zudrop, J., Klimach, H., Hasert, M., Masilamani, K., Roller, S.: A fully distributed CFD framework for massively parallel systems. In: Cray User Group 2012, Stuttgart (2012)