Performance of coupled physics solvers for multidisciplinary hypersonic flow simulations on several classes of computer architectures

David A. Kessler, Andrew Hess, Keith Obenschain,

Laboratories for Computational Physics and Fluid Dynamics

Naval Research Laboratory, Washington, DC 20375

David C. Eder, Alice E. Koniges,

Maui High Performance Computing Center

University of Hawaii

Spencer Starr, Joel Bretheim

General Dynamics Information Technology

Kevin Roe

Boeing

David R. McDaniel, Ryan B. Bond

CREATE Air Vehicles Program

DoD High Performance Computing Modernization Program

Eric J. Nielsen, Aaron Walden, Gabriel Nastac, Kevin Jacobson

NASA Langley Research Center

Anthony Knutson, Graham Candler, Heath Johnson,

Department of Aerospace Engineering and Mechanics

University of Minnesota

Roy L. Campbell

DoD High Performance Computing Modernization Program

The application of hypersonic flow simulation tools to realistic flight scenarios will require the coupling of multiple physical effects to the baseline fluid dynamics. Such multiphysics effects can include the aerooelastic response of the airframe or engine components, dynamic transport of atmospheric particles, the deformation of solid-fluid interfaces that can ablate, pyrolyze, or erode, as well as a host of other processes, all of which are governed by unique sets of physical equations and models. Coupling multiple (and potentially disparate) physics solvers to a robust compressible flow solver poses additional challenges related to the stability, performance and scalability of the combined solver. The choices made during the software design process can therefore lead to a variation in simulation efficiency across different computer architectures. In this paper, we will consider two representative multiphysics hypersonic flow scenarios: the interaction of solid particulates with the flow field created by a hypersonic lifting body and the aerooelastic deformation of a model airframe under high-Mach-number flow conditions. For these simulations we explore the behavior of several hypersonic simulation tools, including Kestrel, FUN3D, US3D, and JENRE® multiphysics framework, on several high performance computing systems containing various CPU and GPU architectures.

I. Introduction

The modeling and simulation of hypersonic flight systems is an inherently multidisciplinary endeavor. The extreme speed associated with this flight regime can impart unique thermal and mechanical loadings on vehicles that must be considered during the design process. Fundamentally, this requires the simulation of a set of physical phenomena governed by their own conservation laws or transport equations that must be fully coupled to the underlying equations governing the unsteady fluid dynamics. There are a number of ways in which this coupling can be accomplished that can be broadly classified into two categories: monolithic solution methods and loosely-coupled approaches. Monolithic solution methods cast multiple physical phenomena into a common set of conservation laws that can be solved using the same numerical integration method. An example of this approach is the unified continuum framework utilized for fluid-structure interaction problems.[1] Loosely-coupled approaches solve potentially different sets of governing equations for each physical process using independent simulation methods. Data is transferred between solvers in the form of source terms, boundary, or initial conditions. While the monolithic approach has a number of inherent advantages, including simplified code design and parallelization strategies, the loosely-coupled approach tends to be preferred in practice. This is due, in part, to the increased flexibility associated with these methods. Solution methods that have been optimized for one set of physical equations can be used in conjunction with those optimized for another.

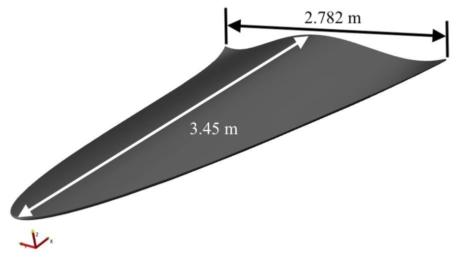
The trade-off that comes along with loosely-coupled solution approaches is the need to communicate information between solvers, which represents both a modeling and a simulation design challenge. From a modeling perspective this requires describing when and where the physics interact, which is an inherently problem-dependent task. In the context of hypersonic vehicle simulations, volumetric source terms can be used to model the body forces imposed on a fluid element by the motion of small-scale particulates in the flow. Likewise, the local fluid state can be used as input to the various models that govern the motion of the particles. For time-dependent simulations, this requires an exchange of body forces and fluid states between the solvers at each physical time step. The mechanics controlling this data transfer constitute one challenge associated with the design of the simulation method. Equally important is the choice of parallelization scheme. Static domain decomposition schemes that attempt to balance work among the multiple computing devices by dividing up the computational domain into sets of nearly equal numbers of cells may be insufficient for these applications since the amount of work does not necessarily correlate with the number of fluid elements. Depending on the particular application, the computational load on each processor can vary drastically throughout the course of a simulation as well, highlighting the importance of dynamic domain decomposition methods that can adjust the computational load on-the-fly.

Resolving the large range of length and time scales and incorporating complex coupled physical processes active in these systems requires the use of large-scale high performance computing (HPC) systems. As the landscape of available computational architectures evolves in the quest for achieving exascale performance, it is important to understand how common multidisciplinary hypersonic simulation tools perform on these next-generation platforms. Recent work by the present authors[2] has shown some variability in the scalability of reacting flow simulations relevant to hypersonic flight systems on various classes of computing architectures. The additional challenges associated with performing multidisciplinary simulations have the potential to further expose differences in performance among these systems. In this paper, we extend these past efforts and focus on two specific multiphysics phenomena: the transport of atmospheric particulates in supersonic and hypersonic flow fields and the aerooelastic response of a model airframe in a hypersonic flow environment. For the simulation packages considered herein, the fluid dynamics solvers are loosely coupled to the solvers used to simulate the additional physics describing particle transport or structural mechanics, The influence of the choice of coupling strategy on simulation performance and scalability will be explored based on the implementations used by four hypersonic simulations tools: the JENRE® multiphysics framework,[3] FUN3D,,[4,5] US3D,[35] and Kestrel.[6–8] The test cases will be run on several different computer architectures, including the Intel® Xeon® processors, AMD EPYC™ processors, and NVIDIA® Tesla V100 GPGPUs.

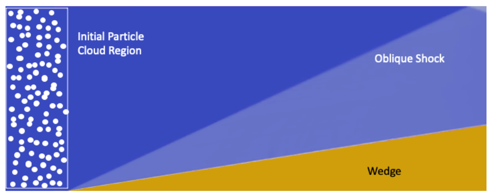
II. Simulation Methods

II.A. Physical Model

The first model system we consider is the external flow over the body of a conceptual hypersonic waverider glide vehicle. This type of vehicle maximizes compression lift by tailoring its outer mold line to match the shape of the shock wave generated by the forebody at the design Mach number. The particular design used in this study utilizes the entire airframe as a lifting surface and is derived using a conical shock analysis.[9] A diagram of the model waverider is shown in fig. 1a. We will consider a single point along a hypothetical flight profile at a speed of



(a)



(b)

Figure 1. Diagrams of the two model problems considered in this study: (a) model waverider geometry and (b) initial particle conditions for the coupled fluid-particle simulations.

Mach 8 and an altitude of 30 km (static pressure and temperature of 1313.4 Pa and 231.66 K, respectively) with the vehicle oriented at zero degree angle of attack. The mechanical loads experienced by the airframe will lead to some deflection across the wingspan, which will, in turn, alter the aerodynamics loading on the airframe. This gives rise to a dynamical system in which the shape and stress on the airframe fluctuate over time.

The second model system that will be considered in this work is the interaction of a cloud of solid particulates with the flow field generated by a simple wedge-shaped lifting body traveling at high-supersonic speeds. This setup is a simplified model of the multiphase flow experienced by a typical hypersonic body interacting with atmospheric particles (either dust particles or liquid droplets associated with cloud formations or precipitation events) during flight through the lower atmosphere. The wedge angle is chosen to be 14 degrees to match the geometry used in other recent studies of solid particulates interacting with supersonic wedge flows.[10,11] In this work, the particulates will be treated as "point" particles with finite masses that can interact with the background fluid in a two-way coupled fashion. We assume a Mach 4 flow at zero degree angle of attack with static temperature and pressure of the ambient air equal to 216 K and 5530 Pa, respectively. Particulates are introduced after the flow has reached a quasi-steady state with the oblique shock attached to the leading edge of the wedge as shown in fig. 1b. The particles convect downstream, interacting with the shock wave and boundary layer prior to potentially colliding with the body surface.

II.A.1. Fluid Dynamics

In both model problems, the flow field is governed by the reacting Navier-Stokes equations for a multi-component gas mixture of N_s species,

$$\frac{\partial C_i}{\partial t} + \nabla\cdot\left(C_i\left(\mathbf{u}+\mathbf{V}_i+\overline{\mathbf{V}}_i\right)\right) = S_{C_i} + \Omega_i, \tag{1}$$

$$\frac{\partial \rho\mathbf{u}}{\partial t} + \nabla\cdot(\rho\mathbf{u}\mathbf{u}+P) = \nabla\cdot(\tau+\overline{\tau}) + S_{\text{mom}}, \tag{2}$$

$$\frac{\partial \rho E}{\partial t} + \nabla\cdot((\rho E+P)\mathbf{u}) = \nabla\cdot\left(\tau\mathbf{u}+\mathbf{q}+\overline{\mathbf{q}}+\overline{\mathbf{D}}_T-\sum_{i=1}^{N_s} W_i C_i h_i \mathbf{V}_i\right) + S_{\text{energy}}, \tag{3}$$

where the total gas phase density $\rho = \sum W_i C_i$ is the sum of the partial densities of the constituent gas species. The corresponding mass fraction of each species is denoted by Y_i. The total energy of the gas phase is the sum of the internal energies of the species ρe_i and the kinetic energy of the gas according to $\rho E = \sum \rho e_i Y_i + 1/2\rho|\mathbf{u}|^2$, where $\mathbf{u}$ is the gas-phase velocity. We assume each species in the gas mixture is an ideal and thermally-perfect gas with an equation of state that is a function of temperature only, $e_i = f(T)$. We use the standard Newtonian viscous stress tensor $\tau = \mu\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T - 2/3\nabla\cdot\mathbf{u}\right)$ and Fourier heat-flux vector $\mathbf{q} = -\kappa\nabla T$, where μ and κ are the mixture-averaged viscosity and thermal conductivity. The species diffusion velocity is computed using a Fickian diffusion law, such that $\mathbf{V}_i = \rho\mathcal{D}_i\nabla Y_i$ and $\mathcal{D}_i$ is a mixture-averaged diffusion coefficient, for each species. The production and destruction of each gas species, Ω_i, is governed by a set of finite-rate chemical reactions, where

$$\Omega_i = \sum_{j}^{N_r} \nu_{ij}\omega_j \prod_{k}^{N_{s,j}} C_k^{b_k}, \tag{4}$$

and ν_{ij} are the stoichiometric coefficients for each of the N_r reactions present in the model and the b_k are the reaction order constants for each species active in a particular reaction. The details of the individual reaction rates, ω_j, are mechanism-dependent but generally take the form of an Arrhenius rate law, $\omega_j = A_j T^{n_j} \exp\left(-T_{A,j}/T\right)$. For the first model problem, we assume that the molecules that comprise the shock-heated air (oxygen and nitrogen) can dissociate and undergo further chemical reactions. The choice of air chemistry model is described separately for each code implementation. For the second model problem, we neglect any chemical reactions and all Ω_i are set to zero. The source terms S_{C_i}, S_{mom}, and S_{energy} represent the exchange of mass, mixture-averaged momentum, and mixture-averaged energy between the gas-phase (composed of three species, oxygen, nitrogen, and water vapor) and the particulates. These source terms will be defined in the following section.

Here, Equations 1–3 have been written generally to represent both the full multi-species reacting Navier-Stokes equations and the Favre-averaged variation of these equations. In the former case, the turbulence quantities, $\overline{\mathbf{V}}_i$, $\overline{\tau}$, $\overline{\mathbf{q}}$, and $\overline{\mathbf{D}}_T$ are assumed to be zero. In the latter case these terms must be modeled in order to close the equations. As in our previous paper,[2] the formulations and models chosen by each simulation group for each test problem will vary and are described in a subsequent section.

II.A.2. Particle Dynamics

Particulates are modeled as point particles governed by the Newtonian laws of motion,

$$\frac{d\mathbf{X}_i}{dt} = \mathbf{V}_i \tag{5}$$

$$m_i \frac{d\mathbf{V}_i}{dt} = \mathbf{F}_i \tag{6}$$

$$\frac{dT_i}{dt} = \frac{\mathrm{Nu}_i}{3\mathrm{Pr}} \frac{\theta}{\tau_i} \left(T - T_i\right), \tag{7}$$

where $\mathbf{X}_i$, $\mathbf{V}_i$, and m_i are the position, velocity, and mass of droplets i. For the flow conditions of interest, we assume that aerodynamic drag is the predominant force acting on the droplets, such that $\mathbf{F}_i = \mathbf{F}_{\mathrm{D}_i}$, where

$$\mathbf{F}_{\mathrm{D}_i} = \frac{1}{2} C_{Di} \frac{\pi d_i^2}{4} \rho \left(\mathbf{u} - \mathbf{V}_i\right) |\mathbf{u} - \mathbf{V}_i|. \tag{8}$$

The droplets are assumed to be rigid spheres with drag coefficient given by the correlation

$$C_D = 0.36 + 5.48\mathrm{Re}^{-0.573} + \frac{24}{\mathrm{Re}}. \tag{9}$$

We note that a variety of drag correlations exist that can more accurately represent particle behavior in high-Mach-number flows. While these models can improve the predictive capability of a simulation of interest, they do not greatly alter the computational cost of the simulations, which justifies the use of the simpler model described above in this effort. The value $\theta = C_p/C_{p,l}$ is the ratio of specific heats of the background gas to the particle material, and $\tau_i = \rho_l d_i^2/(18\mu)$ is the Stokes time constant for the particle. The particle Nusselt Nu_i number follows the correlation: $\mathrm{Nu}_i = 2 + 0.552\mathrm{Re}_i^{1/2}\mathrm{Pr}^{1/3}$ based on the particle Reynolds number and the gas-phase Prandtl number.

II.A.3. Structural Dynamics

The structural dynamics of a linear elastic solid can be written in the general form

$$\mathbf{K}\mathbf{w} + \mathbf{C}\dot{\mathbf{w}} + \mathbf{M}\ddot{\mathbf{w}} = \mathbf{F}_a \tag{10}$$

where $\mathbf{w} = \{d_x \quad d_y \quad d_z \quad \theta_x \quad \theta_y \quad \theta_z\}$ is the vector of displacements and rotations of each node in the structural model. The matrices $\mathbf{K}$, $\mathbf{C}$, and $\mathbf{M}$ represent the material stiffness, damping, and mass matrices, respectively. Solutions for $\mathbf{w}$ can either be obtained exactly through a direct discretization of Equation (10) or approximately by expanding $\mathbf{w}$ as a series of modes, such that $\mathbf{w}(\mathbf{x}, t) = \sum^N \tilde{\mathbf{w}}_i(t)\phi_i(\mathbf{x})$. Equation (10) can then be rewritten as

$$\tilde{\mathbf{K}}\tilde{\mathbf{w}} + \tilde{\mathbf{C}}\dot{\tilde{\mathbf{w}}} + \tilde{\mathbf{M}}\ddot{\tilde{\mathbf{w}}} = \tilde{\mathbf{F}}_a \tag{11}$$

where $\tilde{\mathbf{K}} = \phi^{\mathrm{T}}\mathbf{K}\phi$, $\tilde{\mathbf{C}} = \phi^{\mathrm{T}}\mathbf{C}\phi$, $\tilde{\mathbf{M}} = \phi^{\mathrm{T}}\mathbf{M}\phi$, and $\tilde{\mathbf{F}}_a = \phi^{\mathrm{T}}\mathbf{F}_a\phi$. The mode shapes ϕ_i and associated matrices can be computed *a priori* and stored, giving rise to a set of uncoupled differential equations for the expansion coefficients that must be solved at each time step subject to the aerodynamic forces $\tilde{\mathbf{F}}_a$.

II.B. Implementation Details

II.B.1. FUN3D

FUN3D is a suite of tools developed at NASA Langley Research Center for solving unstructured-grid computational fluid dynamics problems across the speed range from incompressible to hypersonic flows.[4,5,12,13] The solution approach relies on an implicit, upwind finite-volume scheme. Element types may include arbitrary combinations of tetrahedra, prisms, pyramids, and hexahedra, as well as overset mesh paradigms. Control volumes are located at each node and constructed using the median-dual of the grid. Inviscid fluxes are computed at each dual face, or edge midpoint. In this work, the inviscid flux is computed using the HLLE++ scheme,[14,15] with second order accuracy obtained using an unstructured MUSCL reconstruction with unweighted least-square gradients. Viscous fluxes are computed using Green-Gauss element-based gradients, equivalent to a Galerkin-type approximation for tetrahedral elements. For other element types, the gradients are augmented with edge-based data to improve h-ellipticity. To accommodate dynamic and/or deforming mesh scenarios, an Arbitrary Langrangian-Eulerian (ALE) formulation of the governing equations is adopted, where face speeds are included as appropriate and a geometric conservation law is imposed.[4]

For the high-speed flows considered in this work, thermochemical nonequilibrium modeling is used. In this approach, additional governing equations are solved for each constituent species which involve convection, diffusion, and source terms accommodating chemical reactions. A wide range of turbulence closures are available and are discretized in a manner similar to that described above. The governing equations are integrated in time using a fully-coupled implicit dual time-stepping approach. To support the use of distributed-memory architectures, a conventional domain decomposition technique is used with a message-passing approach. Further computational details are described in a subsequent section.

For the aeroelastic analysis demonstrated in the current work, the fluid solver is coupled with a modal structural solver using a staggered method.[4] At each time step, the modal solver computes the deformation of aerodynamic surfaces due to the aerodynamic forces evaluated at the previous time step(s). The volume mesh is deformed to the new surface and the fluid solver is advanced. To conform to the deformed aerodynamic surfaces, the baseline approach used in FUN3D calculates the interior mesh displacements based on a linear elasticity analogy.[4] The computational cost of the modal structural analysis is minimal relative to the fluid solver, but the cost of the linear elasticity procedure can be similar to that of the fluid solver when relatively few temporal subiterations are used. However, because both the modal structural solver and elasticity formulation are linear, the volume mesh displacements at any time step can be computed at a much lower cost using the superposition principle. In this approach, the volume mesh displacements are computed as the sum over the modes of the product of the modal or generalized displacement and the volume mesh displacements due to a unit perturbation of the mode. This is the approach taken in the current work.

FUN3D has been extended to leverage GPU hardware to accelerate the time to solution for both perfect-gas and real-gas applications. The implementation for NVIDIA hardware is based on a CUDA approach, where all kernels required for determining a solution have been ported to execute on the GPU.[2,16–20] In this manner, data transfers between the host and device are minimized and the CPU is used solely for control flow during the solution process. When multiple GPUs are used, a single MPI rank is generally used to shepherd the operations for a single grid partition residing on an individual GPU. MPI communication can be performed using conventional host-based buffers or device addresses as supported by CUDA-aware MPI implementations. Extensive overlapping of communication and computation is used to hide communication latencies. The implementation has been shown to scale well to tens of thousands of GPUs for simulations using meshes containing billions of elements.[21,22]

When solution output is required, asynchronous memory transfers from the device to the host are used to place data directly into asynchronous Input/Output (I/O) buffers on the CPU. In this manner, the overhead associated with writing large amounts of data to disk is completely hidden and does not affect overall computational performance. This approach has been successfully used to produce output data sets consisting of hundreds of terabytes for individual runs.[21,22]

For aeroelastic analysis using the modal structural solver, the volume mesh displacements due to unit perturbations of each mode are precomputed on the CPU once at the beginning of the simulation and stored in GPU memory. This cost is quickly amortized over the course of a typical simulation. At the conclusion of each time step of the fluid solver, the modal structural solver is used to compute the generalized displacements on the CPU, as the cost of this computation is trivial. These values are then copied to the GPU, where the superposition principle is used to determine the new coordinates of the volume mesh.

II.B.2. Kestrel

Kestrel is a simulation tool that integrates physics components together within a flexible, event-based execution paradigm.[7,23] This architecture allows for the intuitive set-up and execution of multi-disciplinary simulations of variable complexity, while allowing for relatively simple modifications to absorb future advances in numerical algorithms and modeling fidelity. Multiple compressible Navier-Stokes flow solvers are currently available for production use with Kestrel; however, solutions for the current work are based on the foundational second-order near-body unstructured solver KCFD. Here, we consider the use of Kestrel for solving dynamic coupled aeroelastic flows,[24] which requires coupling between the underlying fluid solver and a solver that computes the deformation of the body. Two different structural mechanics solvers are available for this purpose: the Sierra/SD solver and modal structural solver.

Kestrels Sierra/SD component interfaces Kestrel with the Sierra/SD Structural Dynamics Solver from Sandia National Laboratories to provide a finite-element-based high-fidelity structural solver. Formerly known as Salinas, Sierra/SD has been shown to scale to problems of 100 million degrees-of-freedom solved on thousands of cores.[25,26] Sierra/SD supports a wide range of elements, including one-dimensional beams and rods, two-dimensional plates and shells, and three-dimensional hexagons and tetrahedrons of various orders. It is capable of computing solutions for a wide range of structural dynamics problems, including vibration and buckling analysis, static analysis, transient analysis (including linear, non-linear, and modal), and various frequency-domain analyses. Kestrel currently only couples with Sierra/SDs linear transient solver, but future work is planned to extend the coupling to include Sierra/SDs non-linear transient solver.

Kestrel's modal structural solver component is responsible for updating the location of the nodes of the modal structural model either by responding dynamically to the surface pressure and viscous forces from the flow solver, or by prescribing certain mode shapes to user-specified values as a function of time. The modal structural dynamics model is a medium- to high-fidelity structural dynamics model that is much less expensive than a full finite-element structural dynamics model solved by Sierra/SD.

When coupling a flow solver with a structural solver, there are two major aspects to the coupling. The first major aspect of the coupling deals with data transfer - specifically, how to impose the forces from the fluid domain onto the structural domain, and how to reflect changes of surface shape from the structural domain into the fluid domain. There are four methods available for the information transfer in Kestrel: a rigid-body attachment, a beam spline, an infinite plate spline, and a thin plate spline. The user tells Kestrel which structural nodes should be mapped to which fluid surface patches, and Kestrel then determines the topology of those structural nodes and chooses the most appropriate of the spline methods. Interpolation matrices are created to transfer viscous and pressure forces from the fluid surface face centroids onto the structural nodes, as well as to transfer the displacements from the structural nodes onto the fluid surface face nodes. The splines constructed are global in the sense that every point in the fluid side of the mapping is influenced to some degree by every point in the structural side of the mapping, resulting in a dense matrix of size $(m \times n \times d)$, where m is the number of points in the fluid side of the mapping, n is the number of points in the structural side of the mapping, and d is the dimensionality of the vector being interpolated (equal to three for displacements and six for forces). In cases where the underlying structural discretization is dense, Kestrel will select a subset of nodes in the structural mapping to include in the interpolation splines. The second major aspect of coupling deals with scheduling, or determining how often the domains are to exchange data. Since both domains are iteratively solved without considering the other domain, a nave interaction can potentially damage the temporal accuracy of the global solution. A predictor-corrector scheme similar to that developed by Farhat[27] that allows for the recovery of second-order accuracy.

After the FSI component has updated the fluid surface mesh node coordinates, the fluid volume mesh is updated to be consistent with the deformed fluid surface mesh. Kestrel performs the volume mesh deformation in two stages.[28] The first stage takes the cells immediately surrounding the surface, called the rigid layer, and deforms them in a rigid fashion in order to maintain cell aspect ratios and other important mesh quality metrics in the viscous region of the mesh. After this, the outer layer is deformed to smoothly absorb the motion of the outer boundary of the rigid layer. Deformation in the outer layer is controlled using a surface influence scheme that describes how much of the mesh deformation is absorbed by translational influence versus rotational influence and decay rates of the mesh motion through the domain.

II.B.3. JENRE® Multiphysics Framework

The JENRE® software suite is a set of multi-physics simulation tools that can provide high-order solutions to flows in complex domains. It encompasses a variety of different finite element solvers, including continuous Galerkin and discontinuous Galerkin formulations in addition to the r-adaptive spacetime MDG-ICE method.[29,30] A variety of conservation laws are supported, including those governing multi-species, chemically-reacting flows, high-speed perfect-gas flows, low-speed incompressible flows, and multi-material (fluid and solid) systems. Additional physics models for coupled discrete particle transport, external electric fields, and surface pyrolysis are also available.

Multiphase flows are simulated utilizing an Eulerian-Lagrangian approach to solve the two-phase flow problem in which the primary and dispersed phases dynamics are computed via a transient time-split phase coupling procedure in which the governing equations are solved by alternately advancing the primary and secondary phase equations over some finite time interval Δt. Data is transferred between the solvers via body forces in the Navier-Stokes equations and interpolation of the fluid state to the location of each of the dispersed particles.

The continuous gas-phase governing equations (Equations (1–3)) are discretized in space using a discontinuous Galerkin (DG) finite element method similar to that described by Hartmann.[31] In this formulation, an approximate Riemann solver (e.g., HLLC, Roe, Lax-Wendroff, etc.) is used to compute the numerical flux across adjacent element faces. The symmetric interior penalty method is used discretize the viscous fluxes, and time integration is done explicitly using a strong stability-preserving Runge-Kutta method. This formulation allows the use of high-order polynomial basis functions, which can increase the formal order of accuracy of the method above second order. We utilize an implicit large-eddy simulation (LES) approach for which the corresponding closure terms in (1–3) are taken to be zero.

The dispersed phase ordinary differential equations are integrated using an explicit three-stage Runge-Kutta method. The size of the primary time step, Δt, is chosen to ensure numerical stability of the gas-phase governing equations with a CFL number equal to 0.05. For the smallest droplets in the system, the time step for the dispersed phase equations can be subcycled so as to provide enhanced numerical stability. Coupling with Equations (1–3) is achieved using a simple cell-based averaging technique in which a filter is used to obtain smooth volume-averaged drag, thermal energy transfer, and vapor mass transfer data at each degree-of-freedom in the Eulerian finite-element mesh,

$$S_Q = \frac{1}{\delta t}\int_t^{t+\delta t} \left(-QK(\mathbf{x}-\mathbf{X}_i)\right) dt \tag{12}$$

for $Q = \{\dot{m}, \mathbf{F}_d, dT_i/dt\}$ where K is a kernel function, taken to be unity in this work. We note that more complex smoothing functions can be used to improve the spatial distribution of the source, and future versions of the solver will incorporate these kernels.

The JENRE® simulation architecture supports both pure distributed memory and hybrid shared–distributed memory parallelism and has been configured to run efficiently on a variety of architectures. On standard CPU systems, shared memory parallelism is achieved using either OpenMP or the Intel® Thread Building Blocks (TBB) library. On GPU systems, support exists for NVIDIA® devices using CUDA and for AMD devices using the Heterogeneous-Compute Interface for Portability (HIP). The JENRE® Mulltiphysics Framework is written in modern C++ and makes heavy use of polymorphism via function and class templates, enabling it to function either as a massively-parallel production code or a cutting-edge research tool. The JENRE® flow solver is used to compute a variety of hypersonic aerodynamic and propulsion systems and is routinely used on both large-scale CPU and GPU systems.

II.B.4. US3D

US3D is an unstructured grid, finite-volume compressible flow solver designed for hypersonic flow applications.[35] Key features of the US3D software include finite-rate chemistry models to accurately simulate thermochemical nonequilibrium and implicit time integration methods to solve the resulting stiff system of equations in a computationally efficient manner. Users have the ability to interface with the US3D software through plugins enabling access to core functionality and allowing

Coupling of the particle and fluid phases is accomplished in two steps. First, the force on each particle (Equation 8) and the heat transfer to each particles is computing using the fluid and particle state at the current time level. The force is computed according to Equation 8 with the Henderson model for the drag coefficient.[32] The heat transfer is computed as to $Q = \pi d_i \kappa (T - T_i)\mathrm{Nu}$ where the Fox model is used to compute the Nusselt number.[33] The particle locations and temperatures are then updated by solving Equations (5)-(7) using first-order Euler time integration. The source terms for the fluid equations are then computed from the particle forces and heat transfer rates. Note that individual particles exchange momentum and energy only with the cell they are contained at the beginning of the time step.

Second, the fluid equations are integrated using first-order Euler implicit time integration with data-parallel line relaxation to solve the linear system.[34] The fluid-particle coupling source terms are included in the fluid equations according to Equations (1)-(3). A constant time step of 50 ns is used to advance the solution. For the current simulation, the particle time step is set to be equal to the fluid time step, however, it is also possible to integrate the particles at a smaller time step than the fluid. Furthermore, higher-order time integration is available in US3D but is not necessary for the present case.

II.C. Computational Architectures

II.C.1. Intel® Xeon® Platinum

The Intel® Skylake Xeon® microarchitecture implements the 512-bit advanced vector extensions (AVX-512) instruction set and has six memory channels per socket. For this study, a representative DoD HPCMP system was used for testing, for which each node is comprised of two Xeon® Platinum 8168 processors with 24 cores per socket and 192GB of memory (96GB per socket). Each core has two AVX-512 512-bit wide single-instruction, multiple-data (SIMD) units. Each node has one Intel® Omni-Path 100Gbps interconnect for inter-node communication.

II.C.2. AMD EPYC^{TM}

The AMD “Zen” architecture is of interest for HPC workloads due to its new design and performance characteristics. AMDs “Zen2” architecture, codenamed Rome, has made numerous changes compared to the initial “Zen” design. There is a dedicated I/O chiplet for memory (eight memory channels per socket) and GMI/xGMI communication with eight chiplets providing up to eight cores per chiplet, for a maximum of 64 cores per socket. There is 32 MB of last level cache per chiplet for a total of 256MB. This is very large compared to earlier AMD and Intel® designs. In addition, the compute capabilities have doubled with two AVX2 256-bit SIMD units per core. A large production system composed of 2,176 dual-socket AMD EPYC™ 7H12 nodes (64 processors per socket).

II.C.3. NVIDIA® Tesla V100

The NVIDIA® Tesla V100 utilizes the Volta microarchitecture, which delivers excellent single and double-precision performance. MPI communication is achieved with the aid of a controlling host IBM Power9 CPU using CUDA-aware MPI. Direct GPU-to-GPU communication across nodes which bypasses host memory is possible through NVIDIA®’s GPUDirect™ remote direct memory access (RDMA) technology using NVLink™ connections. In all cases there is at least one Mellanox 100 Gbps InfiniBand card per two V100 GPUs. Each GPU has 32GB of memory. A cluster composed of 132 V100 devices arranged in a configuration with 6 devices per compute node was used.

III. Results

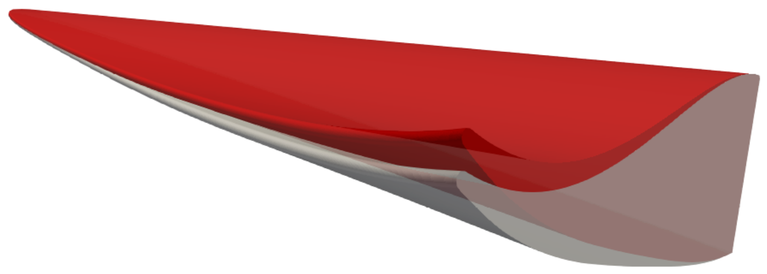


Figure 2. Initial (gray) and final (red) waverider geometries for the static deflection simulation.

The various code groups participating in this study each chose one of the test problems to consider. The FUN3D and Kestrel teams focused on the first test problem (aeroelastic deformation of a hypersonic waverider), and the JENRE® Multiphysics Framework and US3D teams focused on the second test problem (two-phase particle-laden supersonic flow). While the primary focus of this study is to demonstrate how the use of loose coupling strategies for multiphysics simulations affects the parallel performance of the composite solver, it is instructive to provide a basic description of the simulation results and basic physics that drive the solutions that were generated.

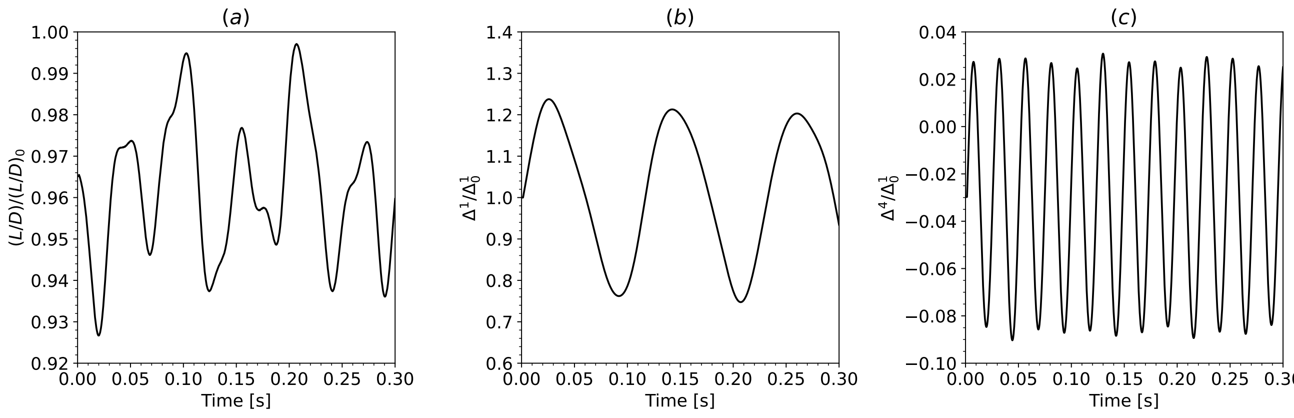


Figure 3. Unsteady aeroelastic waverider results. (a) Lift over drag, (L/D), versus time, normalized by nominal value, $(L/D)_0$, with fixed grid. (b) Generalized displacement for mode one, Δ_1, normalized by static displacement for mode one, Δ_1^0. (c) Generalized displacement for mode four, Δ_4, normalized by static displacement for mode one, Δ_1^0.

For the first test problem, the coupling between the fluid flow and the elastic deformation of the airframe results in noticeable change in the shape of its outer mold line. Two types of analyses are possible. A static analysis provides an equilibrium deformation state for which the displacement of the material points within the solid gives rise to stresses that balance the aerodynamic forces. A dynamic analysis retains the unsteady terms in (10) and provides a time history of the aerodynamic response. Static aeroelastic deflection results for the waverider configuration computed using the FUN3D solver are shown in Fig. 2. The trailing edge tip deflection is approximately 5 cm with the current modeling approach. Dynamic aeroelastic results are shown in Fig. 3. The vertical location of the trailing edge tip oscillates approximately 2 cm around the corresponding location of the static aeroelastic solution. The lift-to-drag ratio is reduced by 4% on average over the nominal fixed-grid value. Generalized displacements for modes one and four are also shown.

For the second test problem, the flow can reach a qausi-steady state in which the shock wave and boundary layers lie in stable, well-defined locations. Density contours through a cross-section of the flow are shown in fig. 4a. Unsteadiness appears in the calculations as the natural response to perturbations to the flow. In this case, such perturbations are generated due to finite resolution effects at the oblique shock as well as perturbations to the flow occur due to the interaction of the secondary-phase particles with the flowfield. The perturbations are transported downstream where they interact with the laminar boundary layer that forms along the surface of the wedge. Ultimately, we are concerned with the rate and angle of particle impact in order to quantify the erosion potential of the particle cloud. An example calculation of several well-defined particle trajectories are shown in fig. 4b. Whether or not a particle impacts the body depends on its in relative proximity to the leading edge, which dictates the amount of time available for the particle to be deflected by the flow behind the oblique shock.

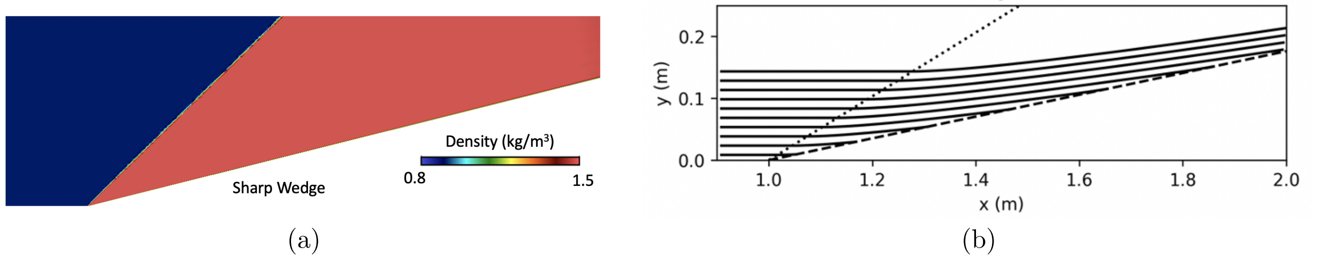


Figure 4. Contours of (a) density and (b) vorticity computed using the JENRE® Multiphysics Framework for the coupled fluid-particle simulation of flow over a sharp wedge.

In the remainder of this section, the seedups presented for each test case and architecture are computed relative to the wall-clock times recorded on one Intel® Xeon® node. We emphasize that each simulation group has utilized their own reference Intel® Xeon® simulation data, and accordingly quantitative comparisons of raw speedup data across simulation tools are not meaningful.

III.A. FUN3D

For the waverider configuration, air is modeled using a 5-species mechanism[36] and the standard one-temperature model. Turbulence is closed using the one-equation Spalart-Allmaras model[37] with Catris-Aupoix compressibility corrections.[38] The vehicle surface is modeled as isothermal and noncatalytic. The flow equations are integrated in time using BDF2 with 5 subiterations per time step. The time step is set to 30 μs, which leads to $\mathcal{O}(100)$ time steps per period for the highest mode frequency. The mesh is composed of 14,545,219 points, 13,222,118 prisms, and 46,515,588 tetrahedra. The surface mesh spacing is 1 cm, with finer 0.1 cm spacing near the planform edges. The location of the first grid point adjacent to the no-slip wall is set to ensure a $y^+ \approx 0.1$ on the vehicle surface. The structural model created for this configuration is a flat plate with uniform thickness and the same planform as the aerodynamic model. The properties of the model are selected such that the deflections of the trailing edge tip are a reasonable value, approximately 5 cm, under the static loads. For the modal solution in the aeroelastic analysis, the first four modes of the model are used. The simulation requires approximately 170 gigabytes of memory at run time.

The aeroelastic simulation is run in three phases. First, a steady state simulation is obtained on the baseline grid. A static aeroelastic simulation is then performed using the baseline solution as an initial condition. Here, a large time step is used to accelerate the convergence to static equilibrium and critical damping is applied to the structural model to ensure it is stable in this phase. Finally, a dynamic aeroelastic simulation is initiated from the static equilibrium solution with a small perturbation applied. The critical damping is removed and the time step is reduced in order to resolve the unsteady physics. The simulation is carried out for three periods of the lowest frequency mode, which corresponds to approximately sixteen periods of the highest frequency mode. Figure 5 depicts strong scaling results on the target architectures for the aeroelastic test problem. Similar results for fluids-only simulations of the same waverider configuration have been previously discussed.[2] Here, an additional expense associated with computation of the grid deformation and accompanying metrics at each time step accounts for a small increase in cost versus the prior results. As such, scaling results are similar to those presented in a previous publication.[2] FUN3D is primarily a memory-bound application; however, it should be noted that the relative performance exceeds the memory bandwidth ratio between the NVIDIA V100 and the CPUs used here. This is because the generic gas CPU implementation is not as highly optimized as its perfect gas counterpart. For perfect gas simulations, GPU speedup is generally commensurate with the ratio of memory bandwidths.

CPU Performance. For FUN3D, a performance advantage of 2.3× is observed for EPYC 7762 relative to Xeon 8168. EPYC 7762 has approximately 1.5× greater aggregate memory bandwidth than Xeon 8168. The speedup beyond the memory bandwidth ratio may be explained by EPYC's higher compute throughput and larger caches. The AVX-512 capability of the Xeon 8168 is not a relevant factor, as FUN3D was compiled to use AVX2 instructions. This approach provided the same or better performance than AVX-512 for the benchmark cases. This is consistent with the general observation that FUN3D has not been thoroughly optimized for generic gas simulations on CPU platforms.

GPU Performance. The absolute speedup of a 6×V100 node over one node of Xeon 8168 is 39.3× for the aeroelastic waverider simulation. The normalized V100 performance relative to Xeon 8168 is 6.6×; that is, a single V100 GPU delivers approximately 6.6× the performance of a dual-socket Xeon 8168 CPU. For the EPYC 7762 node, the 6×V100 node is 17.2× faster, with a normalized V100 performance of roughly 2.9×. An MPI profiling analysis indicates that a slowdown of approximately 1.1× is incurred for the initial 6×V100. Since single-GPU performance is substantially faster than that of the CPU performance, communication overhead has a greater relative impact on GPU performance due to the communication overhead being a fixed cost.

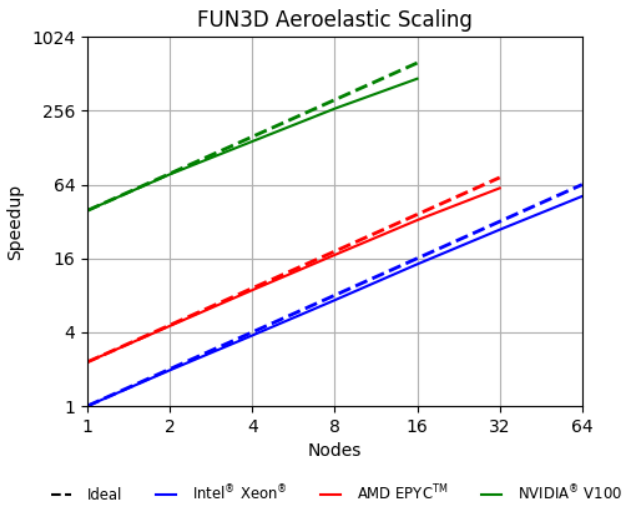
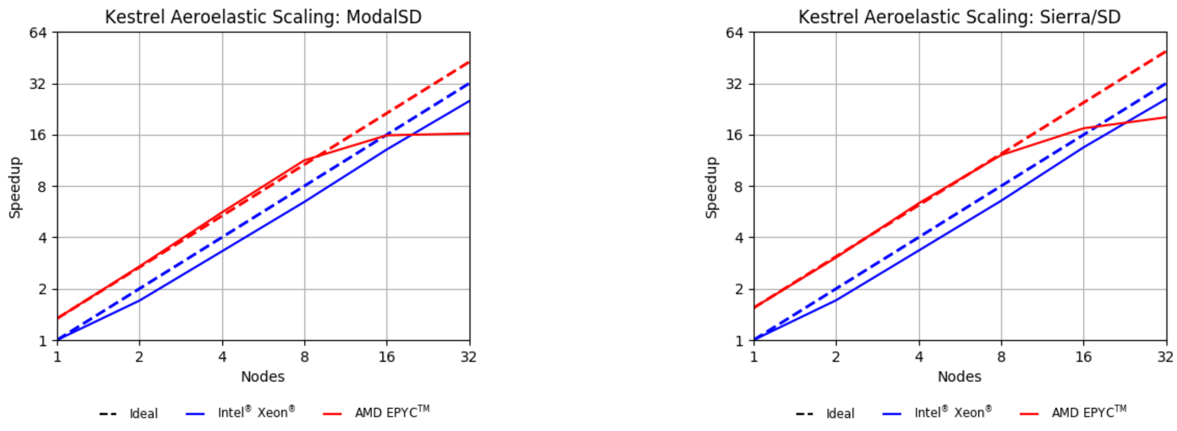


Figure 5. Strong scaling analysis showing speedup relative to the average throughput on a single node of Intel Skylake processors (48 cores) for the aeroelastic waverider test case using the FUN3D flow solver on Intel Skylake (blue), AMD EPYC (red), and NVIDIA Tesla V100 (green) architectures. Dashed lines indicate ideal scaling.

III.B. Kestrel



(a) (b)

Figure 6. Strong scaling analysis showing speedup relative to the average throughput on a single node of Intel Skylake processors (48 cores) for the aeroelastic waverider test case using the Kestrel flow solver on Intel Skylake (blue) and AMD EPYC (red) architectures coupled with (a) the ModalSD solver and (b) the Sierra/SD solver for the structural dynamics. Dashed lines indicate ideal scaling.

Benchmarks for CREATE-AV Kestrel's coupled aeroelastic solvers were run on two representative systems containing either Intel Xeon Platinum 8168 nodes (48 cores/node) or AMD Epyc 7H12 nodes (128 cores/node). Two versions of Kestrel were tested, the production build of Kestrel v12.1 and an optimized build of Kestrel with a new approach for distributing fluid surface data to the structural dynamics solvers. The converged flow field around the static waverider mesh was computed first using the baseline uncoupled KCFD solver. The Menter two-equation turbulence closure model was used, which includes the shear-stress transport (SST) correction for fine-scale effects, the quadratic constitutive relation (QCR) for the Reynolds stress, and the Menter one-equation transition model. This case includes improved delayed detached-eddy simulation (IDDES) terms, resulting in a hybrid RANS/LES model of the flow system. For the context of this analysis, the fluid was taken to be a single-species perfect gas, and accordingly no air chemistry model was employed.

Coupled aeroelastic simulations were started using the steady state gas phase as the initial condition, and the dynamic response of the structure was simulated using the time-accurate predictor-corrector algorithm. Both the native modal (ModalSD) and finite-element (Sierra/SD) structural dynamics solvers were tested. Final wall times used in the analysis were computed as the average solution time per iteration multiplied by the number of completed time steps, which was limited to 50 for these cases. This approach for timing removes overhead from sources such as initialization and output that have no bearing on solver performance. No initial structural deformation was specified for either structural dynamics (SD) package. Runs with both SD packages make use of Kestrel's mesh deformation feature to modify the fluid mesh in response to changes in the structure based on a single rigid layer of elements attached to the waverider body and an outer layer computed using the mesh farfield boundary.

Strong scaling results obtained using both structural dynamics solvers are shown in fig. 6. The parallel efficiency between the two solvers is nearly identical, suggesting that the primary cost for this particular problem resides in the fluid solve and corresponding mesh deformation algorithm. After an initial decrease in speed when moving from one to two nodes on the Intel® Xeon® system, the calculations scaled nearly perfectly out to 32 nodes (1536 cores). On the AMD EPYC™ system, near-ideal scaling was observed out to 8 nodes (1024 cores). Scaling efficiency dropped for the 16 node (2048 core) and 32 node (4096 core) runs. The scaling rolloff observed on Narwhal is unique to this machine. It has been observed with a test code using an MPI communication pattern similar to Kestrel/KCFD and is currently under investigation. The issue is also amplified for this case given the problem size.

III.C. JENRE® Multiphysics Framework

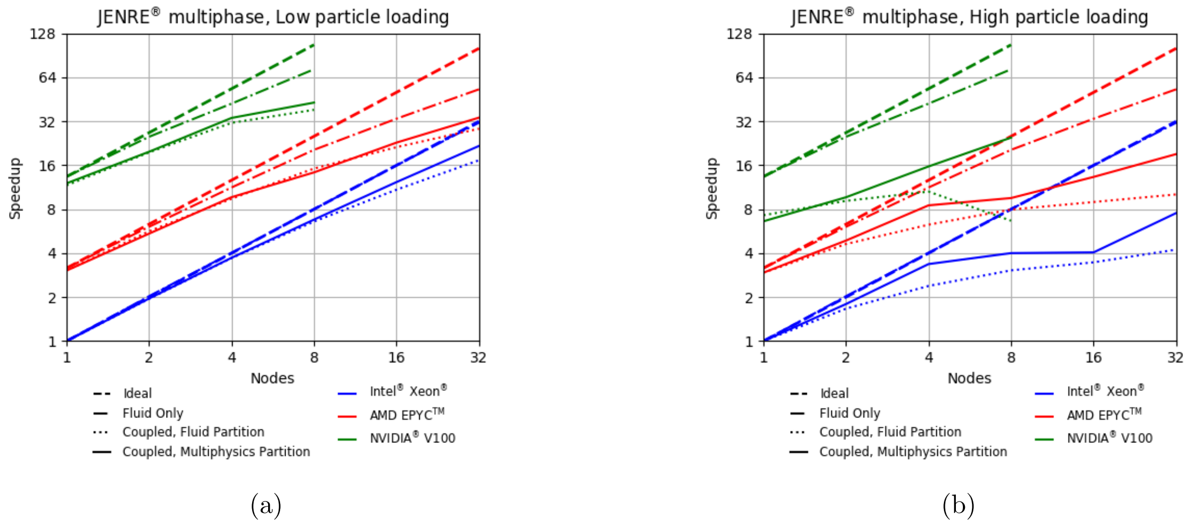


Figure 7. Strong scaling analysis showing speedup relative to the average throughput on a single dual-socket node of Intel® Xeon® processors (48 cores) for the particle-laden supersonic wedge-flow model problem for (a) low particle loading (4.8 million particles) and (b) high particle loading (48 million particles) computed using the coupled fluid-particle solver in the JENRE® multiphysics framework on Intel® Xeon® (blue), AMD EPYC™ (red), and NVIDIA® V100 (green) architectures for two different domain partitioning strategies. Dashed lines indicate ideal scaling. Dash-dotted lines represent scaling for single-phase (fluid-only) simulations.

For the second test problem, supersonic flow enters the domain through a planar face upstream of the body (from left to right as shown in fig. 1) with fixed values of Mach number (4.0), pressure (5530 Pa), species mass fractions (oxygen and nitrogen equal to 0.21 and 0.79, respectively), and temperature (216.7 K). The flow exits the domain through a plane parallel to the inflow plane at a finite distance along the wedge via a characteristic outflow condition. No-slip, adiabatic wall conditions are applied on the surface of the wedge. The computational domain is discretized with an all-tetrahedral mesh that transitions from a body-fitted, anisotropic tetrahedral boundary layer mesh to isotropic elements in the main volume of the flow. Additional mesh resolution is added along the plane of the oblique shock wave anchored to the leading edge of the wedge, giving rise to a mesh containing 13.5 million linear tetrahedral elements.

A single-phase calculation was performed first to obtain a converged flow field to be used in the subsequent multiphase calculations. Timings were gathered for each of the three target architectures, and the associated speedups for these calculations are represented by the dash-dotted lines in Figures 7a and b. For both CPU architectures, we utilized a hybrid distributed-shared memory parallelization strategy with a single MPI rank per node. Shared memory threads were spawned on each node with two threads assigned to each underlying core. While this strategy was found to be sub-optimal for the AMD EPYC$^{\text{TM}}$ system by the authors in a previous study[2] for single-phase fluid calculations, we use it here to reduce the overall communication burden that we will see becomes a bottleneck for our multiphase calculations. We see near ideal scaling on the Intel$^{®}$ Xeon$^{®}$ system (blue) out to 32 nodes (1536 cores). Parallel efficiency drops on the AMD EPYC$^{\text{TM}}$ Rome (red) and NVIDIA$^{®}$ V100 (green) systems. While this is expected on GPU architectures, we note that past scaling results using this flow solver on the AMD EPYC$^{\text{TM}}$ Rome system have indicated better parallel efficiency.[2] using a chiplet-based parallelization strategy.

For the multiphase calculations we consider a particularly challenging case where the particle loading is spatially inhomogeneous. We note that for the companion case of nearly uniform particle loading, the parallel efficiency closely follows that of the single-phase fluid calculations with the only difference arising from the relative communication cost between the two solvers. For the inhomogeneous case, the situation is much different. A domain partitioning strategy based on the single-phase flow solver will result in highly disparate work loads with some MPI ranks receiving the majority of the particles and some receiving few or none of the particles. Also, since MPI communication of particle information is event-based, i.e. data is only transferred when a particle physically crosses a processor boundary in the partitioned mesh, the data transfer load is highly uneven. These observations suggest that a more careful partitioning strategy will be required to achieve good parallel efficiency.

In this section, we compare two different domain partitioning strategies. Both strategies use the Metis (or corresponding ParMetis) libraries, which transform the physical mesh into a graph structure where each element is a vertex of the graph connected to neighboring elements by edges. Graph vertices and edges can be weighted by the relative amount of work and communication overhead associated with each. This first strategy is the naive implementation of the optimized single-phase partition in which the total work (i.e., number of elements) is uniformly distributed among the MPI ranks. Each element (graph vertex) is assumed to require the same amount of work and thus equally weighted in the graph. Processor boundaries are determined by minimizing the overall communication load, characterized by the total number of graph edges that straddle partitions. We refer to the first strategy as the fluid partition. The second strategy attempts to more equally disperse the total computational load on each MPI rank, accounting for both the fluid solve and the number of particles in each cell. This is accomplished by specifying a weight associated with each element (graph vertex). To do this effectively, it is necessary to quantify the relative amount of work required for the fluid solve relative to a single particle update. While this is difficult to estimate *a priori*, it can be measured empirically in a straightforward manner by varying the relative fluid-to-particle cost and measuring the corresponding wall-clock time for a given number of ranks. While the optimal factor will depend both on the problem and the details of the numerical methods used for each solver, we found a relative fluid cost equivalent to 20 particles to be a reasonable approximation. Hence, the total weight assigned to each element was equal to $20 + N_p$ where N_p denotes the number of particles in the element. A new partition was then generated based on these updated element weightings that contains an unequal number of elements per rank. We refer to this particle-weighted partitioning strategy as the multiphysics partition.

Two different levels of particle loading were considered. A “low” loading corresponding to a particle volume fraction of 0.004 and a “high” loading corresponding to a particle volume fraction of 0.04. Note that the particles shown in the digram in fig. 1b only represent a very small portion of the total number of particles in the simulation. For the low particle loading case we transport a total of 4.8 million particles, and that number increases to 48 million for the high particle loading case. Figure 7a shows strong scaling results using the coupled two-phase flow solver for the low particle loading condition. The dotted lines indicate speedups obtained using the baseline fluid partition, while the solid lines indicate those obtained using the multiphysics partition. As expected, the speedups obtained using the fluid partition show a notable drop in parallel efficiency compared to the fluid-only timings (dash-dotted lines) since they were generated without any knowledge of the costs associated with the multiphase flow. For this case, the multiphysics partition was able to recover some parallel efficiency; however the speedups were still far below those obtained for the single-phase simulations. Trends were similar for both CPU and GPU architectures.

Figure 7b shows the strong scaling curves obtained for the high particle loading case. For this configuration, the fluid partition behaved quite poorly for all three architectures considered. Again, this is to be expected given the even greater disparity between the computational loads experienced by each partition. Here, a significant amount of work is associated with the particle solver and MPI ranks containing few or no particles sit idle for an extended period of time. The multiphysics partition significantly improves the situation by more equitably distributing the work associated with the particle solver among the MPI ranks. This is particularly noticeable for the GPU architecture for which throughput is increased by a factor of 3 for the 8-node (48-gpu) case. It should be noted that the GPU results should be interpreted with some care. For the CPU-based systems, the number of MPI ranks is equal to the number of compute nodes used for the calculation. For the GPU system, we use 1 MPI rank per device or 6 MPI ranks per node. Hence the starting position (1 node) for the high particle loading case shows a large decrease in speedup compared to the single-phase simulations since it already includes costs associated with parallelization of the particle solver. Regardless, the slope of the curve obtained using the multiphysics partition is significantly higher than that obtained for the fluid partition and is similar to that obtained for the single-phase simulations.

There are several inferences that can be drawn from this data. The first is that by the nature of the problem, load imbalances can be mitigated but not avoided. Even with the most careful balancing of overall load, MPI ranks will spend time waiting either during the fluid solve or the particle solve (with the exception of a spatially homogeneous particle loading for which the overall load can be balanced for both the fluid solve and the particle solve simultaneously). Second, the localized and inherently dynamic communication load between MPI ranks during the particle solve can have interesting consequences on the scaling, particularly for a relatively small number of total ranks where the additional costs associated with increased communication temporarily outweigh the benefits associated with spreading out the computational load among more ranks. An example of this behavior can be found in Figure 7b. After an initial improvement in scalability going from 1 to 4 CPU nodes, the scaling stagnates. An increase in the number of nodes leads to only marginal speedups in spite of continued spreading of work during the particle solve across more processors. Beyond a certain number of nodes (16 for the Intel® Xeon® system and 8 for the AMD EPYC^{TM}), the scaling once again increases. Note that this behavior is not obvious for the NVIDIA® V100; however, it is possible that this process has already occurred during the partitioning on a single node, which already requires 6 MPI ranks. This suggests that the shape of the partitions may play an equally important role in maximizing scalability. As discussed above, the strategy for defining processor boundaries is to minimize the total edge count along those boundaries. For this class of multiphase simulations, there is also an imbalance in communication during the particle solve for which a careful weighting of the partition graph edges (in addition to the partition graph vertices), could offer additional improvements in scalabitlity.

III.D. US3D

The second test problem described in Section II is simulated with a few differences from the JENRE® test case. A fully hexahedral mesh is containing 50 million cells is used. A no-slip, isothermal wall boundary condition with a constant temperature of 300 K is used along the surface of the wedge while a supersonic outflow boundary condition is used at the exit plane.

Silicon dioxide (SiO_2) particles with a density of 2264 kg/m^3 and a diameter of $1\mu m$ are added at the inflow boundary at a rate such that the freestream mass fraction of particles is 0.0005 or 0.05% resulting in a total of approximate 216 millions particles in the domain at steady-state. At approximate 4.3 particles per grid cell this case is similar to the JENRE® multiphysics high particle loading case with approximately 3.6 particles per cell. Figure 8 flow field is visualized with density contours and the location of every 10,000th particle at an instant in time.

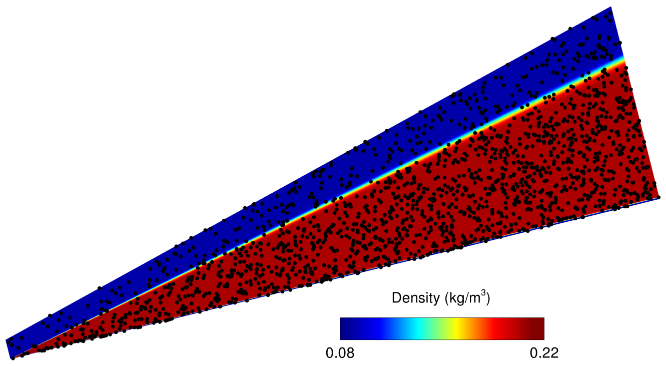


Figure 8. Contours of fluid density and a snapshot of particles locations at steady-state. Note that every 10,000th particle is shown.

Three sets of simulations were run on AMD EPYC^{TM} nodes: pure gas, one-way coupled Lagrangian particle tracking, and two-way coupled fluid and particle simulation. For the one-way particle-only simulations, the pure gas steady-state solution read into memory and particles were injected at the inflow boundary at each time step. The particle motion and temperate was integrated according to the forces and heat transfer rates as previously described. However, the fluid equations were not solved and therefor the fluid state was not updated. For the two-way coupled fluid and particle simulations, the source terms are included in the fluid equations to account for momentum and energy exchange between the particle and fluid. The particle and fluid states are updated at every time step as described in Section II.B.4.

For the present case, the Lagrangian particle tracking portion account for approximate 2/3 of the computational cost each time while the solution to the fluid equations accounted for the other 1/3. The speedup for each set of simulation modes is shown in fig. 9. For the pure gas mode, the scaling is close to ideal up to 32 nodes which is consistent with previous findings.[2] Interestingly, the particle-only simulations exhibit superlinear speedup. The source of superlinear speedup within the parallel particle tracking algorithm is currently unknown but is being investigated and an explanation is forthcoming. The coupled fluid-particle also exhibits superlinear speedup which is to be expected given the particle-only speedup. Overall, US3D shows good parallel efficiency in both the fluid and particle portions of the code. Future work will include more fine-grained timing of the particle movement and parallel exchange routines to explain the speedup observed and identify methods of further decreasing the computation cost and improving parallel efficiency.

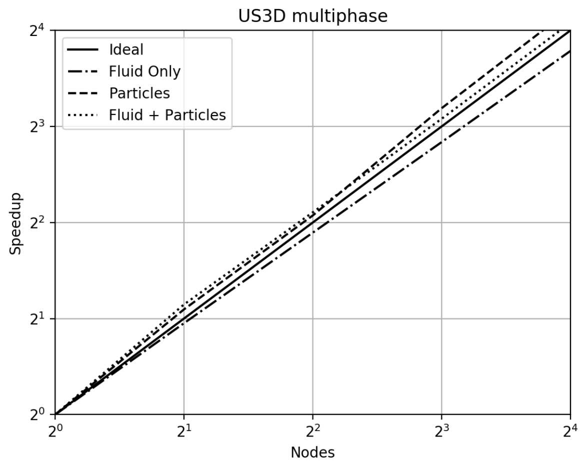


Figure 9. Speedup on AMD EPYC^{TM} nodes.

IV. Conclusions

Two different test problems were considered as part of this effort that were solved using loosely-coupled multi-physics simulation algorithms. In the first test problem, hypersonic flow over an elastic waverider, the two simulation platforms tested (FUN3D and Kestrel) both demonstrated excellent scalability over a range of node counts on both Intel® Xeon® and AMD EPYC^{TM} systems when coupling their fluid solvers to modally-reduced structural solvers. Equally good scalability was observed by the FUN3D team on NVIDIA® V100 GPU devices. Compared to the single-discipline fluid analysis, the majority of the added cost is the deformation of the fluid mesh with this approach. Excellent scalability was also shown on both Intel® Xeon® and AMD EPYC^{TM} systems with Kestrel coupled to a finite-element structure with Sierra/SD. In this configuration the computational load can be balanced independently for both solvers. Coupling between the solvers takes place by exchanging forces and deformations along the material interface between the fluid and airframe. This communication imparts a fixed communication cost on the solvers that can have a small impact on overall parallel efficiency.

In the second model problem that was considered, two physics solvers (Lagrangian particle transport and compressible fluid dynamics) are active in the same regions of the domain and communicate via volumetric source terms and point forces using the JENRE® Multiphysics Framework. A number of unique challenges for large-scale parallel simulations were observed for the case of spatially inhomogeneous particle distributions. Partitioning strategies based on the underlying fluid solver resulted in an unbalanced workload for the coupled multiphysics solver. This caused a severe loss of parallel efficiency on all computational architectures considered. Increasing the particle count from the low to the high loading conditions exacerbates the load imbalance, resulting in even worse performance. A partitioning strategy that weighted each element based on both the computational cost of the fluid solve and the resident number of particles provided some improvement in parallel efficiency, particularly on the NVIDIA® V100 GPU devices. Further improvements will require a more nuanced approach. Additional weights placed on partition graph connections that incorporate some additional knowledge of the physics could prove to be useful. In convection-dominated flows such as the one considered here, more efficient partitions could be generated that weight connections in the general direction of streamlines and hence minimize communication during the particle solve at the expense of increased communication overhead during the fluid solve. Other methods that utilize distinct MPI communicators for the two solvers could also be beneficial. This would allow for independent load balancing of the solvers at the expense of a more complex memory layout and communication strategy.

Acknowledgments

This work was supported in part by the DoD HPCMP, through Naval Sea System Command Award # N00024-19-D-6400, task order N00024-19-F-8814 and Air Force Research Labs Directed Energy Directorate Award # FA9451-20-D-0004, task order FA9451-20-F-0004, to the Maui High Performance Computing Center (MHPCC), which is operated by the University of Hawaii. Computing resources were provided by the DoD HPCMP. The NASA participants in this study would like to thank Bret Stanford from NASA Langley Research Center for generating the waverider structural model and acknowledge the support of the NASA Langley Research Center CIF/IRAD program and the NASA Transformational Tools and Technologies (TTT) Project of the Transformative Aeronautics Concepts Program under the Aeronautics Research Mission Directorate. This material is based upon work supported by, or in part by, the Department of Defense High Performance Computing Modernization Program (HPCMP) under User Productivity, Enhanced Technology Transfer, and Training (PET) contract # 47QFSA18K0111, Award PIID 47QFSA19F0058. Any opinions, finding and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the DoD HPCMP.

References

[1]Idelsohn, S. R., Marti, J., Limache, A., and Oñate, E., "Unified Lagrangian formulation for elastic solids and incompressible fluids: application to fluid–structure interaction problems via the PFEM," *Computer Methods in Applied Mechanics and Engineering*, Vol. 197, No. 19-20, 2008, pp. 1762–1776.

[2]Kessler, D., Johnson, R., Obenschain, K., Eder, D., Koniges, A., Candler, G., Johnson, H., Bretheim, J., Thornburg, H., Roe, K., McDaniel, D., Bond, R., Nielsen, E., Walden, A., Nastac, G., and Campbell, R., "The influence of computer architecture on performance and scaling for hypersonic flow simulations," *2021 AIAA SciTech Forum*, Paper 2021-0144.

[3]Johnson, R. and Kercher, A., "A Conservative Discontinuous Galerkin Discretization for the Total Energy Formulation of the Reacting Navier Stokes Equations," *Journal of Computational Physics*, Vol. 423, 2020, pp. 109826.

[4]Biedron, R. and Thomas, J., "Recent Enhancements To The FUN3D Flow Solver For Moving-Mesh Applications," *47th AIAA Aerospace Sciences Meeting*, Paper 2009-1360.

[5]Nielsen, E. and Diskin, B., "High-Performance Aerodynamic Computations for Aerospace Applications," *Parallel Computing*, Vol. 64, 2017, pp. 20–32.

[6]D.R., M. and Tuckey, T., "HPCMP CREATE^{TM}-AV Kestrel new and emerging capabilities," *2020 AIAA SciTech Forum*, Paper 2020-1525.

[7]Morton, S., McDaniel, D., Sears, D., Tuckey, T., and Tillman, B., "Kestrel A Fixed Wing Virtual Aircraft Product of the CREATE$^{\text{TM}}$ Program," *47th AIAA Aerospace Sciences Meeting*, Paper 2009-338.

[8]Bond, R., Lindorfer, S., and Eymann, T., "Multicomponent and Reacting-Gas Capabilities of HPCMP CREATETM-AV Kestrel v10," *2020 AIAA SciTech Forum*, Paper 2020-1526.

[9]Phoenix, A., Rogers, R., Maxwell, J., and Goodwin, G., "Mach Five to Ten Morphing Waverider: Control Point Study," *Journal of Aircraft*, Vol. 56, 2018, pp. 493–504.

[10]Browne, O. M., Al Hasnine, S. M., and Brehm, C., "Numerical Method for Particulate-Induced High-Speed Boundary-Layer Transition Simulations," *AIAA Journal*, Vol. 59, No. 4, 2021, pp. 1196–1213.

[11]Fedorov, A. V., "Receptivity of a supersonic boundary layer to solid particulates," *Journal of Fluid Mechanics*, Vol. 737, 2013, pp. 105–131.

[12]Anderson, W. K. and Bonhaus, D. L., "An Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids," *Computers and Fluids*, Vol. 23, 1994, pp. 1–21.

[13]Biedron, R. T., Carlson, J.-R., Derlaga, J. M., Gnoffo, P. A., Hammond, D. P., Jones, W. T., Kleb, B., Lee-Rausch, E. M., Nielsen, E. J., Park, M. A., et al., *FUN3D Manual: 13.7*, NASA TM 20205010139, 2020.

[14]Tramel, T., Nichols, R., and Buning, P., "Addition of Improved Shock-Capturing Schemes to OVERFLOW 2.1," *19th AIAA Computational Fluid Dynamics Conference*, Paper 2009-3988.

[15]Nastac, G., Tramel, R., and Nielsen, E., "Improved Heat Transfer Prediction for High-Speed Flows over Blunt Bodies using Adaptive Mixed-Element Unstructured Grids," *AIAA SciTech 2022 Forum*, 2022.

[16]Zubair, M., Nielsen, E., Luitjens, J., and Hammond, D., "An Optimized Multicolor Point-Implicit Solver for Unstructured Grid Applications on Graphics Processing Units," *Proceedings of the Sixth Workshop on Irregular Applications: Architectures and Algorithms*, IA3 2016, IEEE Press, Piscataway, NJ, USA, 2016, pp. 18–25.

[17]Walden, A., Nielsen, E., Diskin, B., and Zubair, M., "A Mixed Precision Multicolor Point-Implicit Solver for Unstructured Grids on GPUs," *Ninth Workshop on Irregular Applications: Architectures and Algorithms*, IA3 2019, 2019.

[18]Nastac, G., Walden, A., Nielsen, E., and Frendi, A., "Implicit Thermochemical Nonequilibrium Flow Simulations on Unstructured Grids using GPUs," *2021 AIAA SciTech Forum*, 2021, pp. 2021–0159.

[19]Stone, C., Walden, A., Zubair, M., and Nielsen, E., "Accelerating Unstructured-Grid CFD Algorithms on NVIDIA and AMD GPUs," *Eleventh Workshop on Irregular Applications: Architectures and Algorithms*, IA3 2021, 2021.

[20]Walden, A., Zubair, M., Stone, C., and Nielsen, E., "Memory Optimizations for Sparse Linear Algebra on GPU Hardware," *Workshop on Memory Centric High Performance Computing*, MCHPC 2021, 2021.

[21]Nastac, G., Korzun, A., Walden, A., Nielsen, E. J., Jones, W., and Moran, P., "Computational Investigation of the Effect of Chemistry on Mars Supersonic Retropropulsion Environments," *AIAA SciTech 2022 Forum*, 2022.

[22]Korzun, A., Nastac, G., Walden, A., Nielsen, E. J., Jones, W., and Moran, P., "Application of a Detached Eddy Simulation Approach with Finite-Rate Chemistry to Mars-Relevant Retropropulsion Operating Environments," *AIAA SciTech 2022 Forum*, 2022.

[23]Morton, S. and Meakin, R., "HPCMP CREATE$^{\text{TM}}$-AV Kestrel Architecture, Capabilities, and Long Term Plan for Fixed-Wing Aircraft Simulations," *54th AIAA Aerospace Sciences Meeting*, Paper 2016-0565.

[24]Lamberson, S., McDaniel, D., and Morton, S., "High-fidelity aero-elastic simulations with HPCMP CREATE-AV Kestrel," *2018 AIAA SciTech Forum*, 2018, pp. 2018–0534.

[25]Bhardwaj, M., Pierson, K., Reese, G., Walsh, T., Day, D., Alvin, K., Peery, J., Farhat, C., and Lesoinne, M., "Salinas: A Scalable Software for High-Performance Structural and Solid Mechanics Simulations," *Supercomputing*, ACM/IEEE 2002, 2002.

[26]Rouse, J. W., Walsh, T. F., and Reese, G. M., "Salinas: A Massively-Parallel Finite-Element Code for Structural Dynamics and Acoustic Analysis," *Journal of the Accoustical Society of America*, Vol. 129, No. 4, 2011.

[27]Farhat, C., Van der Zee, K. G., and Geuzaine, P., "Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aero-elasticity," *Computer Methods in Applied Mechanics and Engineering*, Vol. 195, No. 17, 2006, pp. 1973–2001.

[28]McDaniel, D. R. and Morton, S. A., "Efficient Mesh Deformation for Computational Stability and Control Analysis on Unstructured Viscous Meshes," *47th AIAA Aerospace Sciences Meeting*, 2009, pp. 2009–1363.

[29]Corrigan, A., Kercher, A. D., and Kessler, D. A., "A moving discontinuous Galerkin finite element method for flows with interfaces," *International Journal for Numerical Methods in Fluids*, Vol. 89, No. 9, 2019, pp. 362–406.

[30]Kercher, A., Corrigan, A., and Kessler, D. A., "The Moving Discontinuous Galerkin Finite Element Method with Interface Condition Enforcement for Compressible Viscous Flows," *International Journal for Numerical Methods in Fluids*, Vol. to appear, 2021.

[31]Hartmann, R. and Leicht, T., "Higher order and adaptive DG methods for compressible flows," *VKI LS 2014-03: 37th Advanced VKI CFD Lecture Series: Recent developments in higher order methods and industrial application in aeronautics, Dec. 9-12, 2013*, edited by H. Deconinck, Von Karman Institute for Fluid Dynamics, Rhode Saint Genèse, Belgium, 2014, pp. 1–156.

[32]Henderson, C. B., "Drag coefficients of spheres in continuum and rarefied flows," *AIAA journal*, Vol. 14, No. 6, 1976, pp. 707–708.

[33]Fox, T. W., Rackett, C. W., and Nicholls, J. A., "Shock wave ignition of magnesium powers," *11th International Shock Tubes and Waves Symposium*, 1978, pp. 262–268.

[34]Wright, M. J., Bose, D., and Candler, G. V., "A data-parallel line relaxation method for the Navier-Stokes equations," *AIAA Journal*, Vol. 36, 1998, pp. 1603–1609.

[35]Candler, G. V., Johnson, H. B., Nompelis, I., Gidzak, V. M., Subbareddy, P. K., and Barnhardt, M., "Development of the US3D code for advanced compressible and reacting flow simulations," *53rd AIAA Aerospace Sciences Meeting*, 2015, p. 1893.

[36]Park, C., "On convergence of computation of chemically reacting flows," *23rd Aerospace Sciences Meeting*, 1985, p. 247.

[37]Spalart, P. and Allmaras, S., "A One-Equation Turbulence Model for Aerodynamic Flows," *Recherche Aerospatiale*, Vol. 1, 1994, pp. 5–21.

[38]Catris, S. and Aupoix, B., "Density corrections for turbulence models," *Aerospace Science and Technology*, Vol. 4, No. 1, 2000, pp. 1–11.