AIAA SciTech Forum
11–15 &amp; 19–21 January 2021, VIRTUAL EVENT
AIAA Scitech 2021 Forum

10.2514/6.2021-0144

The Influence of Computer Architecture on Performance and Scaling for Hypersonic Flow Simulations

David A. Kessler, Ryan F. Johnson, 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
Graham Candler, Heath Johnson,
Department of Aerospace Engineering and Mechanics
University of Minnesota
Joel Bretheim, Hugh Thornburg
General Dynamics Information Technology
Gopal Patnaik
Syntek Technologies, Inc.
Kevin Roe
The Boeing Company
David R. McDaniel, Ryan B. Bond
CREATE Air Vehicles Program
DoD High Performance Computing Modernization Program
Eric J. Nielsen, Aaron Walden, Gabriel Nastac
NASA Langley Research Center
Roy L. Campbell
DoD High Performance Computing Modernization Program

It is critical to understand how hypersonic simulation tools perform on a range of computational platforms. This information will aid in the acquisition of appropriate hardware and the potential refactoring of hypersonic codes to run on different systems. In this paper, we consider two representative high-speed reacting flow cases: a model Mach 8 hypersonic waverider glide vehicle and a model hydrocarbon-fueled hypersonic ramjet propulsion system. In both scenarios, the flow fields are in chemical non-equilibrium and are modeled by the multi-species reacting Navier-Stokes equations. For these simulations we use several hypersonic simulation tools, including US3D, Kestrel, FUN3D, and the JENRE® flow solver. We explore several high performance computing systems containing Intel® Xeon® Platinum processors, AMD EPYC™ 7702 processors, and NVIDIA® Tesla V100 devices. We compare performance and strong scaling between the different systems.

DISTRIBUTION STATEMENT A: Approved for public release: Distribution unlimited

I. Introduction

As the global demand for sustained hypersonic flight continues to grow, a renewed research focus aimed at understanding the basic underlying physics that control flight dynamics in this speed regime has emerged. Obtaining the data necessary to perform such analyses is notoriously difficult. Flight tests are expensive and ground test facilities are only able to replicate some of the physical characteristics of actual flight profiles, which leaves numerical simulation as a vital research tool for filling existing knowledge gaps. Resolving the large range of length and time scales and incorporating the 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 hypersonic simulation tools perform on these next-generation platforms.

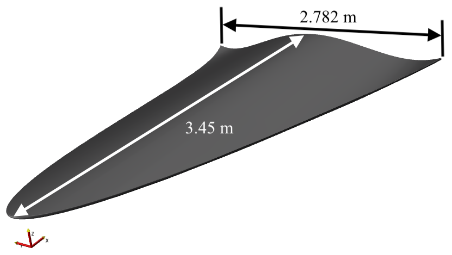
The definition of a hypersonic flow can vary widely depending on the application of interest. For instance, flows inside hypersonic propulsion systems can range from high-subsonic to supersonic speeds with significant heat addition due to exothermic chemical reactions while atmospheric reentry vehicles travel at greater than Mach 20 and produce significant non-equilibrium thermochemical effects in the flow field.[1,2] Hypersonic glide vehicles and hypersonic cruise vehicles, platforms with important defense applications, operate at Mach numbers that can range between 5 and 20 over a range of altitudes and flight profiles. For these systems, thermal and structural material integrity of lifting and control surfaces can often be limiting design factors. Accordingly, in addition to aerothermochemical effects, a variety of other fluid-thermal-structural interactions must be considered.[3] From the point of view of a fluid solver, the net result is a system where the flow is tightly coupled with volumetric source terms driven by combustion and air dissociation chemistry and more loosely coupled to the thermal and structural response of the vehicle via surface interactions. These couplings present a unique set of challenges from a computational science perspective. Herein, we assess how the coupling of non-equilibrium chemistry to a high-speed compressible flow solver affects the overall solver performance. The impact of modeling thermal and structural surface coupling effects will be addressed in a future paper.

The performance (computational speed, memory access, and data communication patterns) of compressible Navier-Stokes solvers has been considered in a number of prior studies that focus on both traditional central processing unit (CPU) and graphical processing processing unit (GPU) architectures.[4–6] Both strong and weak scaling studies indicate efficient performance on those systems. While these studies give a positive indication of the potential scalability of hypersonic flow solvers, for which a compressible flow solver is a basic component, they are insufficient to fully evaluate the challenges associated with performing full vehicle-scale simulations at relevant flight conditions. The diversity of CPU and GPU microarchitectures has also increased in recent years with additional competitors in both the CPU and GPU space. Notably, AMD has recently fielded processors relevant for the high-performance computing space that give some insight into future microarchitecture designs. These processors are designed with multiple "chiplets" on a single unit that provide higher processor yields compared to monolithic designs. This improved computational capacity comes at the cost of increased architectural complexity, however. It is possible that accepted rules-of-thumb for scaling on traditional CPU architectures do not apply on these newer architectures. In recent years, accelerators, such as GPUs, have emerged as strong competitors in the HPC market and are utilized in several of the world's fastest supercomputing systems. Accelerator architectures differ substantially from more traditional CPU architectures and require special programming considerations. This can be even more critical for simulations involving large numbers of gas species that require more memory to store the state of the physical system. In this situation, the optimal memory access patterns could be much different from those used in single-species compressible Navier-Stokes calculations. The coupling of solvers for the chemical production rate equations to the baseline compressible flow solvers can shift the relative cost of computation to communication and fundamentally change the parallel performance. Likewise, certain numerical algorithms could be better suited to specific architectures than others.

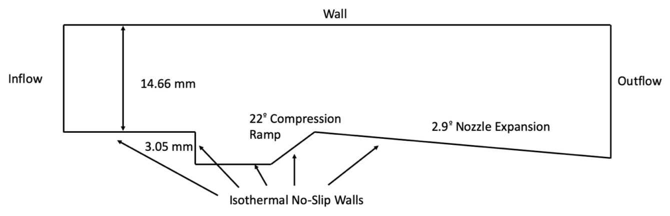
In this paper, we take a step toward answering these questions by analyzing the architecture-dependent computational performance of several hypersonic simulation tools, US3D, Kestrel, FUN3D, and the JENRE® flow solver, on two representative hypersonic chemically-reacting flows. Strong scaling analyses are used to characterize the parallel efficiency on several different computational platforms compared to a baseline architecture, the Intel® Xeon® Platinum 8168 processor. Analysis of the performance of each flow solver between the architectures gives some insight as to the potential for effectively utilizing emerging computational architectures for hypersonic applications.

II. Simulation Methods

II.A. Physical Model



(a)



(b)

Figure 1. Diagrams of the two hypersonic test configurations considered in the present study: (a) model waverider geometry and (b) two-dimensional cross-section of model ramjet combustion facility.

The first model system we consider is the external flow over the body of a conceptual hypersonic waverider glide vehicle, a diagram of which is depicted in fig. 1(a). 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.[7] We consider a single point along a hypothetical flight profile at a speed of 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 second system considered in this study is a model of a hydrocarbon-fueled ramjet combustion facility, the details of which have been discussed by Goodwin et al.[8] A diagram of the computational domain is shown in fig. 1(b). A compressed mixture of ethylene and air enters the combustor from the left and expands into the flame-holding cavity formed by the backward-facing step. The mixture is then accelerated by the compression ramp at the right side of the cavity. The domain then undergoes a mild 2.9 degree expansion before exhausting into a plenum held at atmospheric pressure (not shown in the figure). After ignition, the recirculating flow pattern generated by the flow over the backward-facing step is sufficiently slow to allow a flame structure to develop and provide sufficient energy to stabilize a turbulent flame brush downstream of the cavity.

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,

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

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

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

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, $\omega_j = A_j T^{n_j} \exp\left(-T_{A,j}/T\right)$. Specific reaction models used by each simulation team for the two test problems will be defined in Section III.

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. The formulations and models chosen by each simulation group for each test problem are described in Section III.

II.B. Participating Hypersonic Simulation Codes

II.B.1. US3D

US3D is a highly scalable parallel, unstructured CFD solver, which was developed at the University of Minnesota to predict hypersonic and re-entry flows.[9,10] US3D was originally developed as an unstructured version of the NASA Data-Parallel Line Relaxation (DPLR) code[11] and, as a result, shares many of the same methods and algorithms found in DPLR. However, the use of the data-parallel line-relaxation method[12] on unstructured grids provides greater flexibility when generating grids for complex, three-dimensional vehicles. US3D has successfully been tested and validated against a wide range of high-speed flow applications.

US3D uses a cell-centered finite-volume formulation to solve the compressible Navier-Stokes equations with extensions to account for finite-rate internal energy excitation and chemical kinetics. Convective fluxes are computed using modified Steger-Warming flux vector splitting[13] for steady-state simulations. For unsteady flow applications, low-dissipation centered fluxes are available via multiple spatial flux and time integration methods. Cell-centered gradients are computed using a weighted-least-squares reconstruction of the primitive variables while viscous fluxes are computed using a deferred-correction approach. Rapid convergence to steady state is obtained through the use of the DPLR method along lines of cells normal to surfaces. In regions where lines cannot be formed, a point-relaxation method is used.[14]

US3D includes a large set of built-in surface boundary conditions including adiabatic, isothermal, catalytic and partially-catalytic walls, with and without radiative-equilibrium, wall-blowing and subsonic inflow and outflow conditions. US3D includes capabilities to rapidly converge a perturbed system to a new steady-state using a previously converged steady state solution as an initial starting point. This capability has been used to perform angle-of-attack studies at substantially reduced cost as compared to the standard CFD solver. US3D also includes an extensive set of post-processing routines to help the user understand and visualize their simulation results.

US3D is capable of executing user-defined plug-ins to extend its capabilities which makes it easily extendable beyond just the core software. The US3D API provides an extensive set of entry points for the general solver and post-processor. The user plug-in architecture allows multiple user-written subroutines to be linked into US3D at run-time, which has the effect of allowing US3D to be customized to the modeling scenario of interest.

US3D includes turbulence models for both attached and detached boundary layers. US3D includes multiple turbulence models based upon the Reynolds Averaged Navier Stokes (RANS) formulation and includes an option for the user to add a unique user-defined model via its plug-in capability. These built-in models include the compressible one-equation Spalart-Allmaras (SA) model and the Menter two-equation shear stress transport (SST) model with a vorticity source term (SST-V). In addition, large-eddy simulations (LES) and hybrid RANS/LES simulations can be performed using the high-order numerical methods.

US3D is parallelized using two-level grid partitioning with message-passing. ParMetis[15] is used to partition the grid across nodes, which is then partitioned for the available cores on each node. The partitioning is constrained so that it does not break line-relaxation lines so that the line solves are performed on a single processor. Within the constraints of the methods, asynchronous message passing interface (MPI) communication is used to hide communication with computation.

II.B.2. Kestrel

Kestrel is a production-quality, multi-physics simulation tool targeting engineering solutions for fixed-wing aircraft. It is part of the Computational Research and Engineering Acquisition Tools and Environments (CREATE) Program established in FY08[16,17] by the DoD High Performance Computing Modernization Program (HPCMP). HPCMP CREATE$^{\text{TM}}$ is a tri-service program with a number of unique software products to facilitate a paradigm change from reliance on physical testing as the driver for design iterations to using physics-informed numerical analysis and virtual testing and to serve as a source of actionable engineering data. The core Kestrel design attributes and code capabilities have been well-documented in recent years in numerous conference papers and journal articles (e.g., see references[18–21]).

Kestrel includes three different flow solvers. The one utilized in this current work is referred to as KCFD and employs a finite-volume, cell-centered, unstructured scheme for two-dimensional or three-dimensional discretized domains. The mesh may be composed of tetrahedra, prisms, pyramids, and hexahedra in 3D or triangles/quadrilaterals in 2D. The flow field is solved on a partitioned domain, permitting parallel processing where each partition or zone resides either on a separate processor (PGCore) or shared-memory compute node (PGNode), in which case separate processors share the work for the mesh partition assigned to the compute node. The spatial residual is computed via a typical Godunov scheme[22] with second-order accuracy achieved by the use of linear gradients within each cell. Various approximate Riemann schemes are available to compute the fluxes at each element face, but a slight variation to the HLLE++ scheme[23] is the default and has shown to perform the best over a large range of flow conditions. Up to second-order temporal accuracy is achieved via a subiterative, point-implicit scheme[24] solved using a relaxed Gauss-Seidel technique. Detailed control over the behavior and convergence of the linear problem resulting from the implicit scheme is possible through various advanced inputs.

The one-equation SA turbulence model,[25,26] along with both the baseline[27] and SST[28] variants of Menter's two-equation turbulence model are available to model the fine scale effects of turbulence. The turbulence models have proven accuracy for cases of interest to the DoD fixed-wing community. A detailed description of the formulation with validation cases is given in reference.[29]

Kestrel includes many capabilities that extend beyond perfect-gas air. Perfect gases other than air are supported as well as multispecies, imperfect gases that can be mixing and reacting. Reactions can be handled either with equilibrium or non-equilibrium (a.k.a. “finite rate”) chemistry. Caloric imperfection and temperature-dependent transport properties are handled via curve fits. Thermal imperfection, relevant for some hypersonic ground-test facilities, can also be included via the excluded-volume equation of state. Partial thermal non-equilibrium, relevant for some hypersonic flight conditions and some hypersonic ground-test conditions, can be included via the two-temperature model. Details of these capabilities are given in a series of recent papers.[30–33]

II.B.3. FUN3D

FUN3D is a NASA-developed suite of tools for solving fluid dynamics problems based on the compressible and incompressible Navier-Stokes equations.[34–36] The standard solution approach relies on a spatially second-order accurate node-based finite-volume scheme. Computational meshes may consist of arbitrary combinations of general unstructured element types. At each control-volume face, the inviscid flux is computed using Roe's flux-difference splitting scheme.[37] Solution values at the face are obtained through extrapolation using unweighted least-squares gradients computed at the grid points. The viscous fluxes use the full viscous stress tensors. For tetrahedral meshes, the viscous fluxes are discretized using Green-Gauss element-based gradients, equivalent to a Galerkin-type approximation. For non-tetrahedral elements, the Green-Gauss gradients are augmented with edge-based terms to enhance h-ellipticity. A broad range of turbulence closures are available and are discretized in a manner consistent with the mean flow. Steady-state solutions are obtained using an implicit defect-correction procedure with local time-stepping, where the system of linearized equations is solved using a multicolor point-implicit relaxation. Time-dependent simulations are performed using a temporally second-order accurate dual time-stepping approach in which each physical time step is treated as a quasi-steady-state problem employing the same implicit solution methodology. To achieve a scalable implementation for large distributed-memory systems, a conventional domain decomposition technique is used with a message-passing approach.

Generic gas compositions are accommodated by extending the conservation equations to treat generic gas mixtures and adding additional partial differential equations (PDEs) expressing the convection and diffusion of each constituent species. These additional PDEs are discretized using a fully implicit approach consistent with the mean flow equations. For flows in chemical nonequilibrium, a source term accounts for an arbitrary number of user-specified chemical reactions based on the stoichiometric coefficients for reactants and products and the corresponding forward and backward reaction rates. For high-temperature flows such as those encountered at hypersonic atmospheric entry conditions, thermal nonequilibrium effects may also be considered using a two-temperature model[38] where one temperature describes the translational and rotational energies and the second describes the vibrational, electronic, and electron translational energies. Although included within the current implementation, such thermal effects are not explored in the current work which focuses on lower Mach numbers. For a complete description of the generic gas chemistry approach and implementation, see Gnoffo et al.[39]

The approach described herein has been extended to leverage NVIDIA® Tesla GPUs as a means to accelerate the procedure used to obtain solutions of the governing equations, and has been shown to scale well to thousands of GPUs. In the current implementation, all kernels required for determining solutions are implemented using a CUDA approach.[40–42] 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. 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.[43]

II.B.4. JENRE® Flow Solver

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.[44,45] 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.

The underlying spatial discretization used in this work is based on a conservative nodal discontinuous Galerkin finite-element formulation similar to that described by Hartmann.[46] 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 to 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 spatial flux computations. Source terms are applied using a Strang-splitting approach, and the system of stiff chemical reaction rate laws is integrated implicitly as described by Johnson and Kercher[47] each time step.

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. Currently, CUDA support exists for NVIDIA® GPU devices, and support is being added for AMD GPU devices. The JENRE® flow solver 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.

	Intel® Xeon® Platinum 8168	**AMD EPYC™ 7702**	**NVIDIA® Tesla V100**
Cores/Streaming multiprocessors per socket/device	24	64	80
Sockets/Devices per node	2	2	8
Memory per socket/device	96 GB	128 GB	32 GB
Device/Socket memory technology	DDR4-2666	DDR4-3200	HBM2
Memory bandwidth per socket/device	128 GB/s	204.8 GB/s	900 GB/s
LLC cache size	33 MB	256 MB	6144 KB
Theoretical double-precision performance per socket/device	1920 GFLOPS	2048 GFLOPS	7800 GFLOPS

Table 1. Specifications of the computational platforms considered in this study.

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™

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 cluster comprised of 32 dual-socket AMD EPYC™ 7702 nodes (64 processors) was used with one Mellanox InfiniBand 100Gbps interconnect for inter-node communication.

II.C.3. NVIDIA® Tesla V100

The NVIDIA® Tesla V100 utilizes the Volta microarchitecture, which delivers excellent single and double-precision performance and supports high-speed GPU-to-GPU communication through optional NVLink™ connections. GPU-to-GPU communication across nodes which bypasses host memory is possible through NVIDIA®'s GPUDirect™ remote direct memory access (RDMA) technology and CUDA-aware MPI. In all cases there is at least one Mellanox 100 Gbps InfiniBand card per two V100 GPUs. Each GPU has 32GB of memory.

III. Results and Discussion

As detailed in Section II, the four simulation platforms used in this study vary widely in their choices of numerical algorithms, programming models, and physics models. Accordingly, the preferred strategies for reaching a solution to the two test problems were markedly different. The focus of the analyses presented herein is on the implications of these choices on the observed performance on the different computational architectures. No target validation data is considered, and no attempt is made to compare the relative computational performance or physical accuracy across the four simulation tools. Such analyses are left to the individual groups for future publications. In the remainder of this section, a brief overview of each test problem will be presented along with an illustrative description of the computed flow fields. Each set of simulations will then be analyzed individually by each simulation group and unifying observations related to performance on common architectures will be highlighted as they occur. The speedups 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.

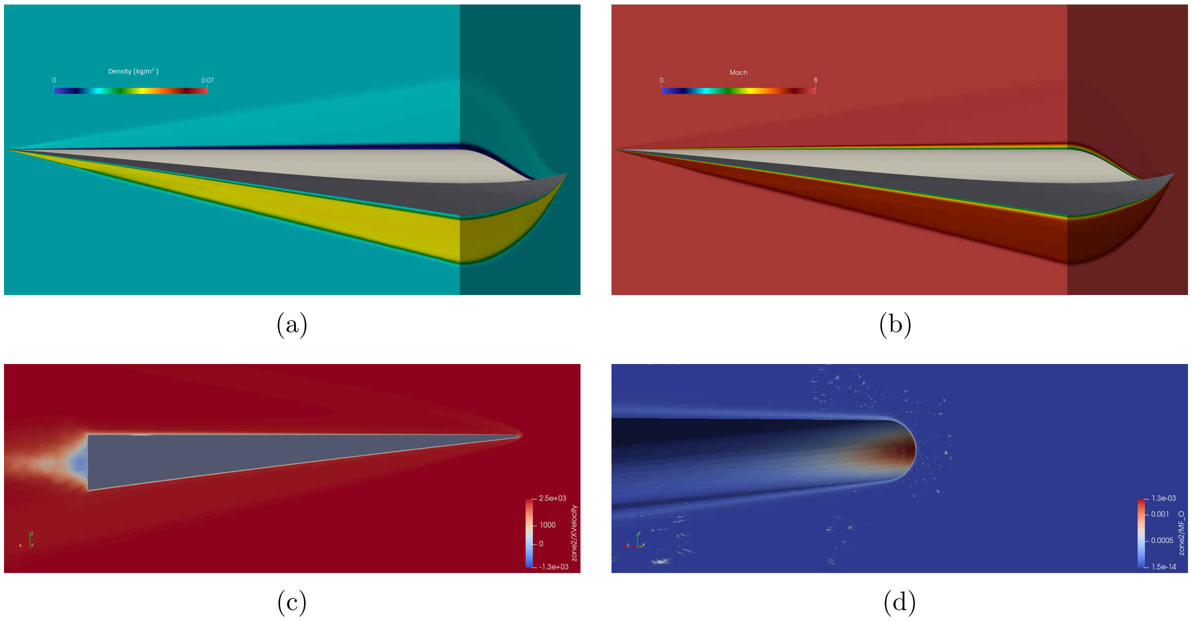


Figure 2. Steady-state flow solutions obtained for the Mach 8 waverider test problem: (a) density and (b) Mach number on planes parallel to and normal to the flow direction computed using FUN3D; (c) streamwise velocity along the center plane and (d) atomic oxygen mass fraction generated near the leading edge of the vehicle computed using Kestrel.

The waverider test problem represents a class of high-Mach-number flight scenarios in which the flow around the body is in a state of thermochemical nonequilibrium, a condition relevant to atmospheric re-entry vehicles and various hypersonic platforms. The flow conditions chosen for this study (Mach 8, 30 km altitude) were relatively mild, however, to ensure that the focus of the analyses remained on computational performance and not on physical modeling. The stagnation temperature for these conditions is approximately 3200 K, which is high enough to generate low levels of oxygen dissociation but low enough to ensure the flow remains in thermal equilibrium. Snapshots of the flow field computed using FUN3D and Kestrel are shown in fig. 2. The gas density contours (fig. 2(a)) highlight the three-dimensional shock structure, which is nearly conical and attached to the outer mold line of the vehicle. The Reynolds number of this flow is quite high (3.2e6 per meter, based on the freestream conditions), resulting in the formation of thin, turbulent boundary layers on the vehicle surface (fig. 2(b)). A strong wake region forms as the flow expands over the sharp trailing edge, characterized by flow recirculation along the base of the vehicle (fig. 2(c)). Near the vehicle leading edge (fig. 2(d)), atomic oxygen is created and transported downstream in the boundary layer where it subsequently recombines into molecular oxygen.

The ramjet combustion test problem represents a class of air-breathing propulsion systems that can be used to power various hypersonic platforms. In this simple configuration, the high-speed flow entering the combustion chamber is slowed by a step expansion in cross-sectional area, which enables stable combustion to occur. Several snapshots of the reacting flow field computed using the JENRE® flow solver are shown in fig. 3. Combustion occurs primarily in the shear layer formed between the high-speed inflow stream and the lower-speed recirculating flow in the cavity of the combustor and in the expanding region of the duct downstream. Figures 3(a) and (b) show contours of the local chemical heat release and mass fraction of an intermediate combustion species (CH_2CO) that highlight the thin, but highly-wrinkled flame front. This

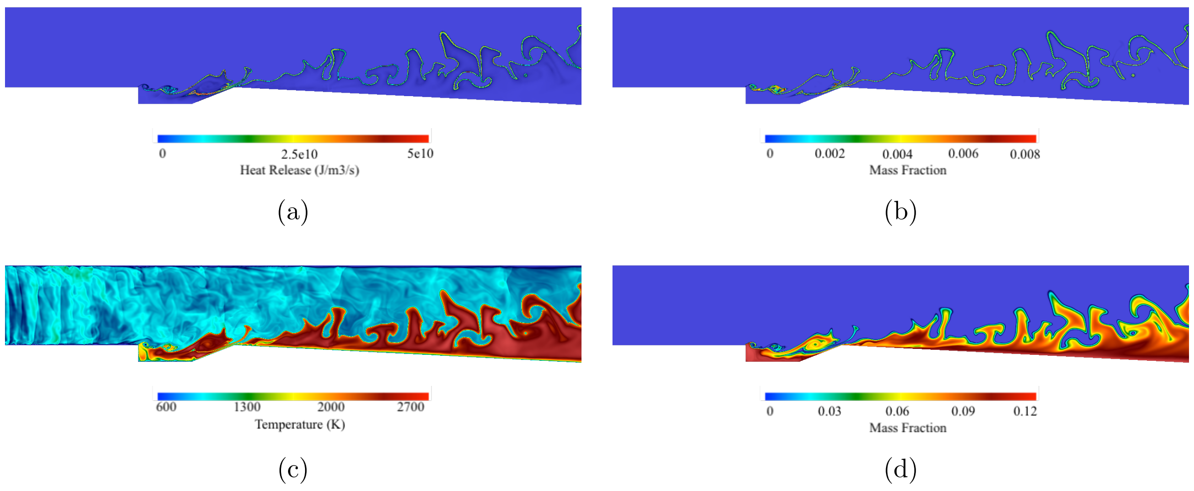


Figure 3. Contours of (a) heat release, (b) mass fraction of CH_2CO, (c) temperature and (d) CO_2 mass fraction at a snapshot in time computed for the cavity-stabilized combustion test problem using the JENRE® flow solver.

flame front is stabilized by a large circulating volume of hot product gases trapped in the cavity region (see the temperature and CO_2 mass fraction contours in fig. 3(c) and (d), respectively). Energy release then continues as the flow travels through the expanding portion of the combustor and generates thrust. This image highlights the localized and dynamic nature of the chemical energy release process that must be captured efficiently by the underlying simulation tools for this test problem. In this case, the influence of free-stream turbulence has been modeled using a synthetic turbulence generation procedure.[48] We note that for a realistic combustor, the structure of the three-dimensional turbulent flame sheet would be more complex.

III.A. US3D Results

The blunt leading edge waverider was computed with both perfect gas and 5-species thermo-chemical nonequilibrium reacting air. The baseline cases assume a laminar boundary layer; a radiative equilibrium, non-catalytic surface boundary condition ($\epsilon = 0.89$) is used represent the waverider surface. NASA 9-coefficient polynomials were used for the thermodynamics and a standard Park two-temperature model was used for the chemical kinetics. The transport properties were computed from Blottner curve fits and Wilke mixing rule; Fickian diffusion was used for the mass diffusion.

The geometry and near wake were represented with a 15.364 million element hexahedral grid; the grid was generated with LINK3D.[49] The grid has a near-wall spacing such $y^+ < 1$. Time integration is performed with the data-parallel line relaxation method and the fluxes are represented with second-order accurate modified Steger-Warming flux vector splitting. The CFL number was ramped from an initial value of 1 to a maximum value of 20,000 over about 200 time steps (CFL ramping factor of 1.05); at CFL $= 20,000$, $\Delta t = 16\,\mu$s. Larger time steps can be used, but there is little effect on the solution convergence. The L_2 density residual decreases by five orders of magnitude in about 1100 time steps; the drag is converged to with 0.1% of the final value in 448 time steps, and with 0.01% in 1346 time steps. For this case with low levels of chemical reactions, the perfect gas simulation has similar convergence properties to the 5-species case. If the cases had been run with RANS turbulence models active, the computational costs for 5-species air would increase by approximately 20% for Spalart-Allmaras and 44% for the Menter SST model. There is minimal effect on the rate of convergence to steady-state because the solution of the turbulence transport equations is tightly coupled to the fluid conservation equations.

The AMD EPYC$^{\text{TM}}$ and Intel® Xeon® scaling results are plotted in fig. 4. Here, the results on both machines are scaled relative to the computational cost on a single node of the Intel® Xeon® system. Timings were performed by restarting from a converged solution and running for 100 time steps. The average compute time per time step is reported. For the AMD EPYC$^{\text{TM}}$ system, all cases were run with 128 cores per node. For reference, the single-node timing for the 5-species gas model is 4.798 s on the AMD EPYC$^{\text{TM}}$ system

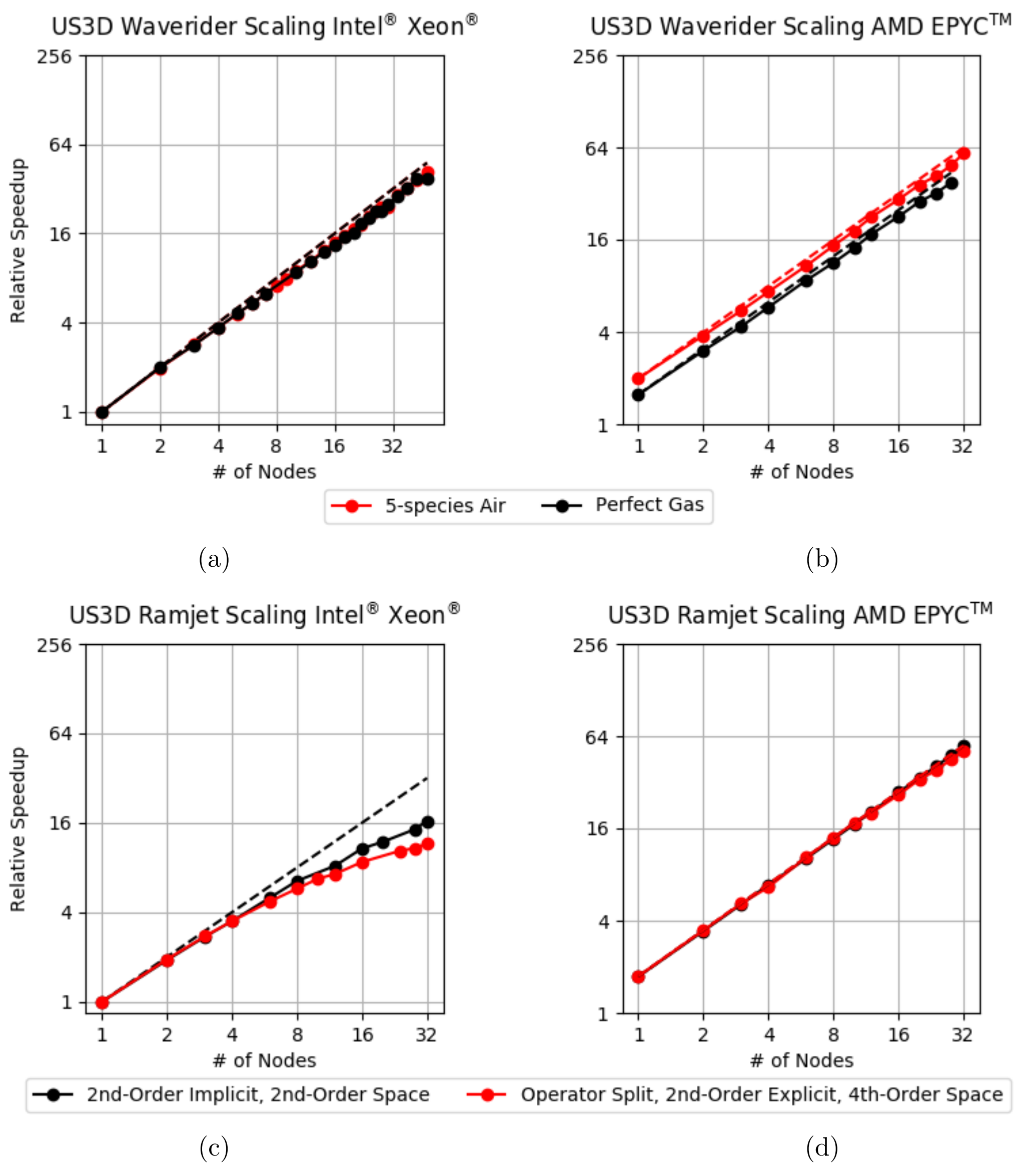


Figure 4. Strong scaling analysis showing speedup relative to the average throughput on a single dual-socket node of Intel® Xeon® processors (48 cores) for the (a, b) waverider and (c, d) ramjet combustion test case using the US3D flow solver on Intel® Xeon® (a, c) and AMD EPYC^{TM} (b, d) architectures using several different physics models and numerical methods. Dashed lines indicate ideal scaling.

and 11.419 s on the Intel® Xeon® system; thus, the AMD EPYC™ system is 2.38 times faster than the Intel® Xeon® system for US3D running on a single node. For the perfect gas laminar simulation, each time step requires 1.451 s on the AMD EPYC™ system and 2.264 s on the Intel® Xeon® system; a factor of 1.56 speed advantage.

The scaling results show reasonably good performance on both systems. For the 5-species case, the AMD EPYC™ system has a speedup relative to one node of 13.5× on 16 nodes (84%) and 24.5× on 32 nodes (77%). The Intel® Xeon® system produces speedups of 13.7× on 16 nodes (86%), 26.2× on 32 nodes (82%), and 37.4× on 48 nodes (78%). the AMD EPYC™ system's superior computational performance exposes the communication inefficiencies relative to the Intel® Xeon® system. For reference, 16 nodes of the AMD EPYC™ system is equivalent to 40 nodes of the Intel® Xeon® system.

The scaling of the perfect gas calculations is better than the 5-species case. For the AMD EPYC™ system, the speedup is 15.11× on 16 nodes (94%) and 28.5× on 32 nodes (89%). The Intel® Xeon® system's speedup is 13.3× on 16 nodes, 26.8× on 32 nodes, and 39.7× on 48 nodes, all of which correspond to 83% efficiency. This is somewhat surprising given that the perfect gas simulation requires significantly less computational work; this may be offset by improved cache performance. For this gas model, 16 of the AMD EPYC™ system nodes are equivalent to 28 the Intel® Xeon® system nodes.

The perfect gas scaling results discussed here were obtained using a version of US3D that is compiled specifically for single species perfect gas calculations. If the general gas version is used, there is some reduction in compute performance, and this penalty is significantly larger for the Intel® Xeon® system than the AMD EPYC™ system. The single node compute time goes from 2.264 s to 2.927 s on the Intel® Xeon® system; on the AMD EPYC™ system it is 1.451 s and 1.474 s. It appears that the Intel® compiler is able to better optimize the perfect gas code. Another series of runs were completed on the AMD EPYC™ system using a US3D binary compiled with all native compiler optimizations, including AVX2 vectorization instructions. The performance obtained using this binary was marginally worse than that obtained using the baseline optimization level for all three thermodynamic models.

A University of Minnesota AMD cluster was also used to run scaling studies. This machine has the same AMD 7702 processors as the AMD EPYC™ system used by all the groups in this study, but its memory performance (DDR4 2933 MHz) is less than that of the common system (DDR4 3200 MHz). It uses a single Mellanox Infiniband switch running at 200 GB/s. The waverider runs are about 13% slower on this cluster relative to the AMD EPYC™ system, which is in close proportion to the memory performance. This is expected because US3D is highly memory intensive.

The ramjet benchmark case was run with a 20-species C_2H_4-Air model.[50] The Vreman subgrid-scale turbulence model was used to represent the effects of unresolved turbulence scales. However, the results of the ramjet simulation are not expected to be completely realistic because we are using a two-dimensional model of the full system. The grid has 3.83 million hexahedral elements representing the duct, cavity and dump tank. A fixed mass flux and temperature characteristic inflow boundary condition was used. The thermodynamics and transport properties are represented with the same models as used for the waverider.

Turbulent combustion simulations typically require the resolution of unsteady turbulent motion and the stiff chemistry associated with the combustion reactions can limit the stable explicit time step. Thus, the numerical methods required for the ramjet are very different than for the waverider. We ran the ramjet with four different methods, ranging from a low-order upwind implicit method to a high-order operator-split method tailored for stiff chemical source terms. These studies show that the single-node AMD EPYC™ system performance is between 1.73× and 2.21× times that of the Intel® Xeon®-based system. Furthermore, the AMD EPYC™ system scales better than the Intel® Xeon® system for these test cases. For example, for the fully implicit time integration method, the AMD EPYC™ system shows excellent scaling with a speedup of 15.8× on 16 nodes and 31.8× on 32 nodes; the Intel® Xeon® system's speedup is just 10.8× and 16.5×, respectively. When using an operator-split integration technique (second-order in time, fourth-order in space), the speedups obtained on the AMD EPYC™ system were similar at 15.5× and 30.8×; on the Intel® Xeon® system the speedups drop to 8.8× and 11.7×. These methods require very different computation and communication.

For the ramjet, the relative performance of the systems is very significant. For example, on the Intel® Xeon® system one time step of the implicit method requires 0.867 s on 32 nodes – the same performance can be obtained on the AMD EPYC™ system with 10 nodes. For the high-order operator split method, this measure is even more compelling, with just 5.2 nodes of the AMD EPYC™ system producing the same performance as 32 nodes of the Intel® Xeon® system.

III.B. Kestrel Results

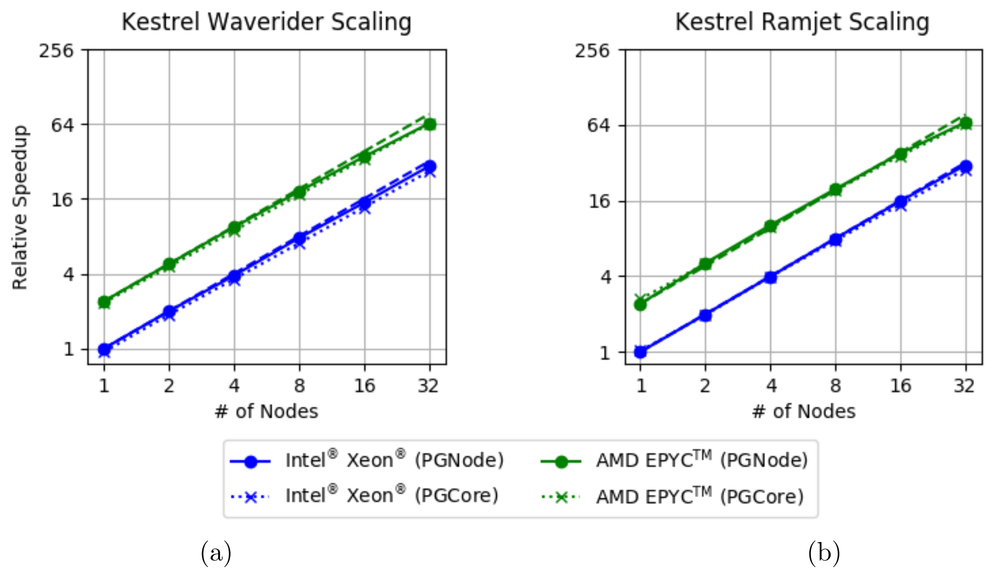


Figure 5. Strong scaling analysis showing speedup relative to the average throughput on a single dual-socket node of Intel® Xeon® processors (48 cores) for the (a) waverider and (b) ramjet combustion test case using the Kestrel flow solver on Intel® Xeon® (blue) and AMD EPYC™ (green) architectures. Solid lines obtained using node-based partitioning (PGNode), and dotted lines obtained using core-based partitioning (PGCore). Dashed lines indicate ideal scaling.

While Kestrel v10.4.1 offers three distinct flow solvers, we use the default option (KCFD) for all cases reported herein. Both the flameholder and waverider cases solve the compressible turbulent Navier-Stokes equations set with second-order accuracy. Other solver configuration details differ between the two cases and are described in this section.

The waverider case solves the Menter two-equation turbulence model for the RANS calculations and 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. The gas is specified as a mixture in chemical non-equilibrium with the 5-species, 9-equation Park model.[38] The boundary conditions are a symmetry plane at the body midplane with no-slip, adiabatic walls on the body with a Mach 8.0 flow at static pressure and temperature of 1313.4 Pa and 231.66 K, respectively. The benchmarking timing period is 100 iterations with four sub-iterations per iteration. The time integration is accomplished with a local time-stepping scheme at a CFL of 1.0 and with advective temporal damping of 0.025. The mesh for this three-dimensional simulation case contains 14,389,695 cells. As in the flameholder case, the cells per zone ranges widely, from as little as 3,235 (PGCore, 32 nodes on the AMD EPYC™ system) to as high as 14,389,695 (PGNode, single node on the Intel® Xeon® system).

The flameholder case, which models ethylene combustion in a ramjet configuration, solves the Spalart-Allmaras (SA) one-equation turbulence model for the RANS calculations. The gas is specified as a mixture in chemical non-equilibrium with a 20-species, 35-equation chemistry model.[50] At the upstream boundary, the boundary conditions are uniform inflow (source) at Mach 0.6, total pressure of 174,279 Pa and 1125 K total temperature. The downstream boundary is an exit plenum (sink) region at atmospheric pressure. The ramjet walls are no-slip isothermal walls at 1100 K. The benchmarking timing period is 500 iterations with four sub-iterations per iteration. The time integration is accomplished with a global specified time-stepping scheme with a timestep of 5.0e-8 s and advective temporal damping of 0.025. The geometry and mesh for this two-dimensional simulation case are based off those used in the paper by Johnson et al. and contains 1,523,013 cells. In Kestrel, the mesh is partitioned into a number of zones, with the number of zones equal either to the number of processors (PGCore) or to the number of nodes (PGNode). As such, the number of mesh cells per zone ranges widely, from as little as 364 (PGCore, 32 nodes on the AMD EPYC™ system) to as high as 1,523,013 (PGNode, single node on the Intel® Xeon® system).

When viewed in a per-processor context, there is minimal difference between the strong scaling performance of KCFD for these two cases on the Intel® Xeon® and AMD EPYC$^{\text{TM}}$ systems. Across both cases and both mesh partitioning schemes, strong scaling efficiencies range from 87% to 98% in the 1500-2000 core range (Intel® Xeon® and AMD EPYC$^{\text{TM}}$) and from 78% to 87% at 4096 core (AMD EPYC$^{\text{TM}}$ only). The general rule of thumb for KCFD is to try and keep the mesh density above 10,000 elements per core. This guidance is justified by the current results in that the efficiencies here stay above 90% under these conditions. A marked benefit is shown with per-node mesh partitioning (PGNode) in the flameholder case. Both the Intel® Xeon® and AMD EPYC$^{\text{TM}}$ results showed about 10% better efficiency at the highest core counts relative to per-core mesh partitioning. Super-linear speedup is observed up to 1024 core (8 nodes) on the AMD EPYC$^{\text{TM}}$ system for this case. This is assumed to be due to a combination of the larger number of operations per cell being accomplished due to the complex chemistry, the potential for more of the element arrays to fit fully in cache due to the small mesh size, and the larger core/node ratio on the machine which reduces the MPI bottlenecks in the per-node partitioning scheme. Regarding relative performance between the two systems tested with Kestrel (again, normalized to a per-processor context), the AMD EPYC$^{\text{TM}}$ system was about 11% and 22% faster on 1 and 32 nodes, respectively, for both cases when using node partitioning. When using core partitioning, the AMD EPYC$^{\text{TM}}$ advantage was 4% and 16% faster, respectively for the flameholder case and 8% and 11% faster, respectively, for the waverider case. Interestingly, the speed advantage of the AMD EPYC$^{\text{TM}}$ system over the Intel® Xeon® system was at a minimum for both cases in the 2-4 node range. Finally, we note that the Kestrel builds used in this study were compiled using the same strategy as typical production versions of the software, which focuses on providing a robust and portable simulation platform. No performance tuning beyond the use of the predefined generic compiler optimization flags was attempted, although efforts are underway to explore this possibility.

III.C. FUN3D Results

For the waverider configuration, air is modeled using a 5-species mechanism[51] and the standard one-temperature model. Turbulence is closed using the one-equation Spalart-Allmaras model[52] with Catris-Aupoix compressibility corrections.[26] The vehicle surface is modeled as isothermal and noncatalytic. The equations are integrated in time using steady-state local time stepping. The mesh is composed of 15,004,094 points, 13,447,413 prisms, 11,922 pyramids, and 48,178,239 tetrahedra. The location of the first grid point adjacent to the no-slip wall is set to ensure a $y^+ \approx 1$ on the vehicle surface. The simulation requires approximately 170 GB of memory at run time.

The ramjet chemistry is modeled using a 20-species, 35-reaction ethylene mechanism [50] with the standard one-temperature model. An explicit turbulence model is not used, resulting in an implicit Large-Eddy Simulation (ILES) approach. A wall spacing of $y^+ \approx 1$ based on the freestream Reynolds number is used. The walls are modeled as isothermal and noncatalytic. Ignition is performed by initializing a small region of radius 0.002 m at 2500 K downstream of the cavity lip. The final mesh is derived through a simple prismatic extrusion of a two-dimensional triangular mesh and is composed of 3,379,488 points and 3,373,895 prisms. While the conventional FUN3D implementation offers a two-dimensional solution procedure, the current GPU implementation is restricted to three dimensions. For this reason, all ramjet simulations have been performed as three-dimensional cases for consistency. The simulation requires approximately 175 gigabytes of memory at run time.

Figure 6 shows strong scaling results for FUN3D on the target architectures. FUN3D scaling is determined primarily by three components. Foremost is the linear solver described in Section II.B.3. The solver exchanges partial halo information after each color group and employs a heuristic number of sweeps through the linear system, which in practice results in $\mathcal{O}(200)$ partial halo exchanges to recover the serial algorithm. This communication, generally implemented using MPI, is overlapped with the computation of interior solutions. The communication scales linearly with the number of unknowns while the computation scales quadratically, which results in near-perfect scaling for the 20-species ramjet in all cases and similar for the 5-species waverider, though the scaling worsens at 32 nodes of the Intel® Xeon® system.

FUN3D scaling is also affected by MPI communication in its residual computation. For simulations of generic gas flows, there are three halo exchanges of gradients and flux limiter information as well as a global norm of the residual vector. The exchanges may be overlapped in a manner similar to that of the linear solver;[40] however, overlapping is only implemented for one of the gradient exchanges at this time. Finally, scaling may be affected by load imbalances introduced by the mesh partitioner, which can be exacerbated on meshes involving multiple element types which incur disparate costs. This effect is generally no worse

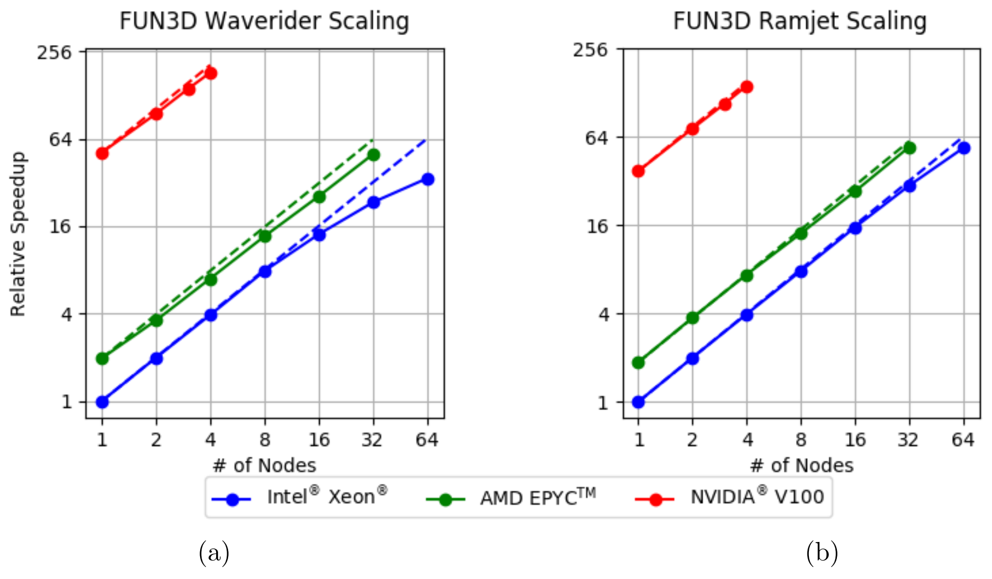


Figure 6. Strong scaling analysis showing speedup relative to the average throughput on a single dual-socket node of Intel® Xeon® processors (48 cores) for the (a) waverider and (b) ramjet combustion test case using the FUN3D flow solver on Intel® Xeon® (blue), AMD EPYC™ (green), and NVIDIA® V100 (red) architectures. Dashed lines indicate ideal scaling.

than 5%.

FUN3D is primarily a memory-bound application; however, it should be noted that the relative performance for V100 exceeds the memory bandwidth ratio between V100 and the various CPUs. 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.

For FUN3D, we observe a performance advantage of 1.8× for the AMD EPYC™ processor relative to the Intel®Xeon® processor. The AMD EPYC™ processor has approximately 1.5× greater aggregate memory bandwidth than the Intel® Xeon® processor. The speedup beyond the memory bandwidth ratio may be explained by the AMD EPYC™'s higher compute throughput and larger caches. The AVX-512 capability of the Intel® Xeon® processor is not a factor as FUN3D was compiled to use AVX2 instructions, which provided the same or better performance than AVX-512 for the benchmark cases. This is explained by the lack of thorough optimization of FUN3D's generic gas CPU path, mentioned elsewhere in this section.

The absolute speedup of an eight-way NVIDIA® V100 node over one node of the Intel® Xeon® system is 51.4× for the waverider case. The normalized NVIDIA® V100 performance relative to the Intel® Xeon® is 6.43×; a single V100 device delivers 6.43× the performance of a dual-socket Intel® Xeon® node. Based on previous single device comparisons, we would expect a number closer to 7.5 for this case, but the smallest V100 result employs 8 GPUs due to the memory requirements of the case. Profile analysis indicates that an approximate slowdown of 1.12× is incurred for the initial 8-GPU result. This is a result of non-overlapped MPI communication in the residual, load imbalance, and a poor transfer rate from the system PCIe bus, which was on average only 7 GB/s using page-locked buffers. Communication costs are essentially fixed, so they have a greater relative impact on the faster GPUs. As the application to Intel® Xeon® CPUs is scaled up, however, the normalized single device performance increases to a factor of 12.08× for the NVIDIA® V100. Given the scaling trend for the Intel® Xeon® processors, it is unlikely that any number of nodes on that system could deliver more absolute performance than a single node of eight NVIDIA® V100s for this case.

The absolute speedup of an 8-way NVIDIA® V100 node over one node of Intel® Xeon® processors is 37.3× for the ramjet case. The normalized NVIDIA® V100 performance relative to a dual-socket Intel® Xeon® node is 4.66×. This number is lower than that of the waverider for several reasons. The ramjet mesh is a pseudo-2D mesh composed entirely of prismatic elements. This configuration results in a nearly equal number of elements and nodes. For typical 3D mixed-element cases, such as the waverider, this ratio is closer to five. The GPU speedup beyond the memory bandwidth ratio for FUN3D's generic gas capability is a result of the left-hand side matrix construction being highly optimized for GPU and relatively poorly optimized for CPU. Because the number of cells is 5× lower than the waverider case, the left-hand side occupies a significantly reduced portion of the overall execution time. The overall speedup is thus more reflective of the speedup of the linear solver, which is close to the memory bandwidth ratio.

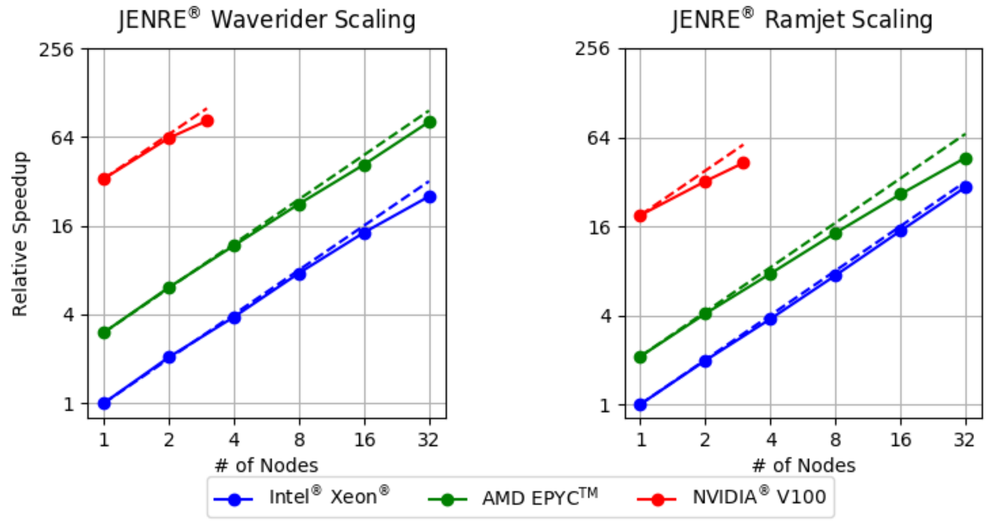
III.D. JENRE® Flow Solver Results

The JENRE® software suite provides an interesting test case for this analysis. The underlying spatial discretization is formulated using the discontinuous Galerkin (DG) finite element method, which uses piecewise polynomial basis functions to represent the flow field in each computational element. While the derivations of both the DG and finite volume methods are based on the weak form of the conservation equations, the DG method retains the volumetric integral terms for basis functions with polynomial degrees of $p \geq 1$. We note that the use of piecewise constant ($p = 0$) basis functions results in a discretization that is similar to a finite volume discretization. These additional volumetric terms can provide better spatial accuracy for the flow solution on equivalent size grid cells, but they add to the overall computational cost of the simulation and change the underlying memory-access patterns. Accordingly, this type of algorithm could be sensitive to the design of the underlying computational architectures on which it is compiled.

We begin by considering the performance on the first test problem (cf. fig. 1(a)): Mach 8 flow over the model waverider geometry. In this set of calculations, supersonic flow enters the domain through a planar face (y-z plane) upstream of the body with fixed values of velocity (2500 m/s), pressure (1313.4 Pa), species mass fractions (oxygen and nitrogen equal to 0.21 and 0.79, respectively), and temperature (231.66 K). The flow exits the domain through a plane parallel to the inflow plane two body-lengths downstream of the body via a characteristic outflow condition. No-slip, isothermal wall conditions ($T = 500$ K) are applied on the surface of the vehicle. We consider only half of the vehicle body and apply symmetry conditions along the $y = 0$ plane. 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. The total element count is 24,641,059 tetrahedra. Park's 5-species model[38] is used to simulate the dissociation and recombination chemistry. No explicit turbulence closures are used, and the resulting calculations can be classified as implicit LES.

Ideally, the boundary-layer mesh would be constructed to ensure that the height of the first cell layer represents $y^+ \approx 1$ (computed using an equivalent intra-element spacing based on the number of degrees of freedom within the element). For this high-Reynolds-number flow, the effective grid wall-normal spacing would be $\Delta x \approx \mathcal{O}(10\mu\text{m})$. The implication of this requirement is that this flow is best suited for solution using an implicit time-advancement method, since stability restrictions for explicit time-advancement methods would require an exceedingly-large number of time steps to reach a steady solution. For this analysis, we have chosen to sacrifice spatial resolution in order to reach a converged solution in a reasonable length of time, and in all subsequent calculations, the grid was designed to provide a $y^+ \approx 20$. Regardless, the solver exercises all of the physical models and solution algorithms of interest to this study and the measured computational performance and scaling data are representative of what would occur with increased spatial resolution. For this test problem, we utilize piecewise constant ($p = 0$) polynomial basis functions. The use of higher-order basis functions will be considered in the context of the second test problem.

Strong scaling results for the waverider test problem are shown in fig. 7(a) for the Intel® Xeon®, AMD EPYC™, and NVIDIA® V100 architectures. As with all earlier analyses, the relative speedup is measured with respect to the compute time on a single Intel® Xeon® node. The computational domain is divided into roughly equal-sized partitions, with the Intel® simulations utilizing one partition per node, the AMD EPYC™ simulations utilizing one partition per non-uniform memory access (NUMA) domain (8 partitions per physical node), and the V100 simulations utilizing one partition per individual GPU device (8 partitions per node). For the AMD EPYC™ simulations, threads were bound to each NUMA domain (16 threads per partition), while no thread binding was specified for the Intel® simulations (48 threads per partition). Both of the CPU architectures exhibited near-ideal scaling out to 16 nodes and eight nodes on the Intel® Xeon® and AMD EPYC™ systems. On the Intel® system, a pronounced dropoff is seen in the 32-node data. On the AMD EPYC™ system, however, the deviation from ideal is less pronounced. Comparing the system performance on a per-core basis (i.e., dividing the node-based speedup data by the ratio of AMD EPYC™ cores per node to Intel® cores per node, 2.667), the AMD EPYC™ system performed roughly 10% better per core for the 1 through 16 node simulations. At 32 nodes, the ratio increases to roughly 20%. While the baseline 10% deviation could be attributed to unoptimized choice of thread placement on



(a) (b)

Figure 7. (a) Strong scaling analysis showing speedup relative to the average throughput on a single dual-socket node of Intel® Xeon® processors (48 cores) for the (a) waverider and (b) ramjet combustion test case using the JENRE® flow solver on Intel® Xeon® (blue), AMD EPYC™ (green), and NVIDIA® V100 (red) architectures. Dashed lines indicate ideal scaling.

the Intel® nodes, the widening gap at the higher node count could have important implications for code scalability on these systems. A similar trend can be seen in the FUN3D scaling results as well; however, we temper this observation by noting that the scaling results obtained for the ramjet test problem (cf. fig. 7(b)) show contradictory results. Additional data on a larger AMD EPYC™ system will be needed to draw any definitive conclusions.

We next consider the computational performance measured for the second test problem: ethylene combustion in a high-speed propulsion system (cf. fig. 1(b)). From a physical standpoint, this problem differs from the waverider test problem in the size and complexity of the chemical kinetics mechanism used to compute the reacting flow field. While a number of reduced models for ethylene-air combustion exist, in this work we utilize a 19-species, 35-reaction step elementary skeletal mechanism[50] to investigate how increasing the number of conserved variables affects the overall computational performance of the solver on the three test architectures. In order to limit the overall memory requirements associated with performing this simulation, we restrict our simulations to two dimensions and discretize the cross-section shown in fig. 1(b). The left boundary is a subsonic inflow of a homogeneous ethylene-air mixture at Mach 0.6, 1.72 atm, and 1125 K, with a fuel equivalence ratio of 0.42. The wall boundary conditions are no-slip and isothermal. The top wall and bottom wall upstream of the cavity are set to 800 K, the cavity walls are set to 1000 K, and the expanding portion of the nozzle is set to 1100 K as measured in Nielsen et al.[53] The nozzle discharges into a large plenum region with non-reflecting outflow boundaries at atmospheric pressure. We use an isotropic triangular mesh with varying spatial resolution, ranging from 15 μm in the cavity shear layer, flame, and against the walls to 110 μm in the core flow. While the overall total of 1,091,014 elements used in this simulation is much smaller than the total number of elements used in the waverider test case, we consider higher-order spatial discretizations ranging from $p = 1$ (3 DOFs per element) to $p = 3$ (10 DOFs per element). The total number of degrees of freedom (DOFs) in the simulations thus range from 3,273,042 ($p = 1$) to 10,910,140 ($p = 3$). Furthermore, the total number of conserved variables per DOF for these simulations is 22, compared to nine for the three-dimensional waverider simulations. These simulations exercise the full complexity of the DG method, requiring both cell-based and face-based residual evaluations. The overall cost of the chemical kinetics source term is also greater for this test case which consists of 35 rate equations compared to nine for the Park 5-species air chemistry model. As with the waverider simulation, the implicit LES approach is used for these calculations, and explicit turbulence models are not considered.

A strong scaling analysis is provided in fig. 7(b) for the ramjet test case. In this figure, all data was generated using linear ($p = 1$) polynomial basis functions. Interestingly, the data suggest two opposite trends. Scaling on the AMD EPYC™ and NVIDIA® V100 architectures is noticeably worse than for the waverider scaling analysis. This can be explained by examining the total amount of memory required for this test case. For this $p = 1$ example, the total memory usage for this test case is lower than that required for the waverider test case. The increase in the number of DOFs per element and the number of conserved scalars is offset by the large reduction in total elements used in this analysis. It is unsurprising then that the scaling deviates from ideal at a lower node count as the total amount of work is divided into smaller chunks. One surprising observation is that the trend appears to be opposite on the Intel® Xeon® platform. For this case, near ideal scaling is observed out to 32 nodes, which represents an improvement over the results obtained for the waverider test case. It is worth reinforcing the fact the the total amount of compute required per DOF in this case is significantly greater than that for the waverider test case, however. The net effect of these competing factors could lead to the observed increased sensitivity to details of the thread partitioning strategy.

Figure 8(a) shows the effect of simultaneously increasing both the number of DOFs in each partition and the amount of compute performed for each DOF by increasing the polynomial degree of the underlying basis functions. The total number of residual calculations for the $p = 3$ case is roughly equivalent to the $p = 0$ waverider calculation. For all three platforms, the scaling improves with increasing p. Additionally, fig. 8(b) shows the single node speedups obtained on the AMD EPYC™ system using node-based (single partition per physical node, unbound threads) to the NUMA-based partitioning strategy (single partition per NUMA domain, threads bound to their respective NUMA domains) for each test problem. For both cases, the NUMA partitioning strategy was more efficient than the node-based strategy; however, a sizable gap in performance was found for the waverider test case. While a number of factors could contribute to this, it is likely that the large increase in the ratio of compute to memory access for the ramjet test problem results in a diminished sensitivity to the memory layout.

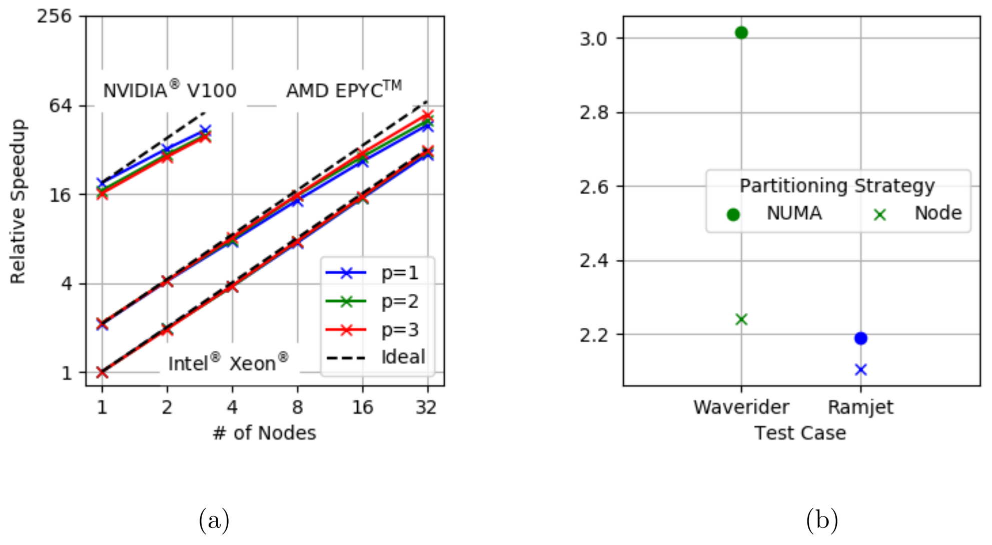


Figure 8. (a) Speedup relative to the average throughput on a single dual-socket node of Intel® Xeon® processors (48 cores) computed with the JENRE® flow solver using three different spatial discretizations with polynomial basis function of degree $p = 1$ (blue), 2 (green), and 3 (red). (b) Single-node speedups obtained on the AMD EPYC™ architecture using two different partitioning and thread binding strategies: 1 MPI rank per node with unbound threads ('x' symbols) and 1 MPI rank per non-uniform memory access (NUMA) domain with threads bound to each domain ('o' symbols).

We close this section by considering the relative code performance on the four-node (32-device) V100 system used in this study for the two test problems. In both cases, a significant speedup was obtained using this architecture: roughly 33x per node (4.1× per device) for the waverider test problem and 19x per node (2.4× per device) for the ramjet test problem. For the former, the speedup is close to the memory bandwidth ratio between the V100 and Intel® Xeon® architectures. The ramjet calculations exhibited an overall lower speedup and less than ideal scaling. For the waverider test problem, however, reasonable scaling was achieved on two nodes.

IV. Conclusions

In this study, we have examined the computational performance of four hypersonic simulation tools, US3D, Kestrel, FUN3D, and the JENRE® flow solver, on three computational architectures of relevance to existing and near-term HPC architectures. The computations focused on two classes of chemically-reacting flow fields that are of relevance to research and development efforts within the DoD and the broader scientific community: high-Mach-number flows in chemical non-equilibrium and high-speed combustion systems for hypersonic propulsion applications. While each simulation tool exhibited unique scaling behavior on the various architectures on which it was tested, several overarching conclusions can be drawn.

In terms of CPU performance, the AMD EPYC™ nodes tested in this study gave speedups relative to the Intel® Xeon® nodes across all four simulation platforms for both of the test problems. The measured speedups ranged from approximately 1.8× to 3.0× for the various test cases. These factors are, in all cases, larger than the memory bandwidth ratio between the two processors (1.6×) but, for the majority of cases, less than the ratio of cores per socket (2.667×) between the processors. While additional profiling studies are needed to provide a definitive answer, it is likely that larger caches and the increased number of cores available on the AMD EPYC™ processors are responsible for the observed speedups. An additional observation is that all four simulation tools were portable across the Intel® and AMD platforms with no additional compiler-specific tuning required. We note, however, that the performance results also suggest that the various simulation platforms are not making full use of the SIMD capabilities available for either architecture. Optimizations to improve vectorization for AVX2 and AVX-512 instruction sets will most likely improve performance for all codes on both the AMD and Intel® platforms. While the AMD EPYC™ processors did provide better performance for the test problems considered in this study, we note that the Intel® Xeon® processors that were utilized were part of an in-production HPC machine and were subject to complicating factors, such as network contention with other scheduled jobs. The AMD EPYC™ processors, on the other hand, were part of a dedicated test system.

In terms of GPU performance, the two simulation tools that can utilize the NVIDIA® V100 GPU system (the FUN3D and JENRE® flow solvers) exhibited a degree of variability in the speedups obtained for this architecture. For both groups, relative speedups were found to be lower for the ramjet test problem, most likely caused by the reduction in grid size and dimensionality for that test case. We also point out that even for the cases that exhibit relatively good scaling, that some decrease in parallel efficiency is masked by the normalization with respect to a single *node* of NVIDIA® V100 devices. Both groups observed a dropoff in efficiency when spreading across devices within a single node. An interesting observation can also be made regarding the waverider test problem for which both code groups observed a marked decrease in parallel efficiency on the Intel® Xeon® system. The dropoff was sufficiently large that achieving equivalent performance on the Intel® system compared to a single node of eight V100s would require a very large number of Intel® Xeon® nodes. Furthermore, good parallel efficiency was also observed when the work was spread across several V100 nodes, increasing the gap in performance at large CPU node counts.

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 Division 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. Additional support was also provided through DoD HPCMP's Productivity Enhancement and Training (PET) program. Computing resources were provided by the DoD HPCMP. The NASA participants in this study would like to 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. We also thank Dr. Gabriel Goodwin (Naval Research Laboratory) for providing the ramjet combustion test case.

References

[1]Bertin, J. and Cummings, R., "Critical hypersonic aerothermodynamic phenomena," *Annual Review of Fluid Mechanics*, Vol. 38, 2006, pp. 129–157.

[2]Gnoffo, P., Weilmuenster, K., Hamilton, H., and Venkataphathy, E., "Computational aerothermodynamic design issues

for hypersonic vehicles," *Journal of Spacecraft and Rockets*, Vol. 36, 1999, pp. 21–43.

[3]Culler, A. and McNamara, J., "Studies on fluid-thermal-structural coupling for aerothermoelasticity in hypersonic flow," *AIAA Journal*, Vol. 48, 2010, pp. 1721–1738.

[4]Turpault, R. and Nguyen-Bui, T.-H., "A high order MOOD method for compressible Navier-Stokes equations: application to hypersonic viscous flows," *Progress in Computational Fluid Dynamics, an International Journal*, Vol. 19, No. 6, 2019, pp. 337–345.

[5]Di Renzo, M., FU, L., and Urzay, J., "HTR solver: An open-source exascale-oriented task-based multi-GPU high-order code for hypersonic aerothermodynamics," *Computer Physics Communication*, 2020.

[6]Terrana, S., Nguyen, C., and Peraire, J., "GPU-accelerated Large Eddy Simulation of Hypersonic Flows," *AIAA SciTech 2020-1062*, Paper 2020.

[7]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.

[8]Goodwin, G., Johnson, R., D.A., K., and Chelliah, H., "Premixed Ethylene-Air Combustion in a Dual-Mode Scramjet Cavity Combustor with a Turbulent Inflow," *2019 AIAA Propulsion and Energy Forum*, Paper 2019-4270.

[9]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.

[10]Johnson, H. B., Drayna, T. W., Nompelis, I., and Candler, G. V., "US3D Software User Manual," *Department of Aerospace Engineering and Mechanics of Univeristy of Minnesota, version*, Vol. 1.

[11]Wright, M. J., White, T., and Mangini, N., "Data Parallel Line Relaxation (DPLR) Code User Manual: Acadia-Version 4.01. 1," 2009.

[12]Wright, M. J., Candler, G. V., and Bose, D., "Data-parallel line relaxation method for the Navier-Stokes equations," *AIAA Journal*, Vol. 36, No. 9, 1998, pp. 1603–1609.

[13]MacCormack, R. W., *Numerical computation of compressible and viscous flow*, American Institute of Aeronautics and Astronautics, Inc., 2014.

[14]Wright, M. J., Candler, G. V., and Prampolini, M., "Data-parallel lower-upper relaxation method for the Navier-Stokes equations," *AIAA Journal*, Vol. 34, No. 7, 1996, pp. 1371–1377.

[15]"ParMETIS - Parallel Graph Partitioning and Fill-reducing Matrix Ordering," `http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview`.

[16]Post, D., Arevalo, S., Atwood, C., Bell, P., Blacker, T., Dey, S., Fisher, D., Fisher, D., Genalis, P., Gorski, J., Harris, A., Hill, K., Hurwitz, M., Kendall, R., Meakin, R., Morton, S., Moyer, E., Strawn, R., van Veldhuizen, D., Votta, L., Wynn, S., and Zelinski, G., *Journal of Physics: Conference Series*, Vol. 125, 2008, pp. 012090.

[17]Meakin, R., Atwood, C., and Hariharan, N., "Development, Deployment, and Support of a Set of Multi-Disciplinary, Physics-Based Simulation Software Products," *49th AIAA Aerospace Sciences Meeting*, 2011, pp. 2011–1104.

[18]McDaniel, D. and Tuckey, T., "Multiple Bodies, Motion, and Mash-Ups: Handling Complex Use-Cases with Kestrel," *49th AIAA Aerospace Sciences Meeting*, Paper 2014-0415.

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

[20]McDaniel, D., Tuckey, T., and Morton, S., "The HPCMP CREATE$^{\mathrm{TM}}$-AV Kestrel Computational Environment and its Relation to NASAs CFD Vision 2030"," *55th AIAA Aerospace Sciences Meeting*, Paper 2017-0813.

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

[22]Godunov, S., "A Finite Difference Method for the Numerical Calculation of Discontinuous Solutions of the Equations of Fluid Dynamics," *Matematicheskii Sbornik*, Vol. 47, 1959, pp. 271–290.

[23]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.

[24]Tomaro, R., Strang, W., and Sankar, L., "An Implicit Algorithm for Solving Time Dependent Flows on Unstructured Grids," *35th Aerospace Sciences Meeting and Exhibit*, Paper 97-0333.

[25]Spalart, P. and Allmaras, S., "A One-Equation Turbulence Model for Aerodynamic Flows," *30th Aerospace Sciences Meeting and Exhibit*, 1992.

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

[27]Menter, F. R., "Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications," *AIAA Journal*, Vol. 32, 1994, pp. 1598–1605.

[28]Menter, F. R., Kuntz, M., and Langtry, R., "Ten Years of Industrial Experience with the SST Turbulence Model," *Turbulence, Heat and Mass Transfer 4, ed*, edited by H. K., N. Y., and T. M., Begell House, Inc., 2003, pp. 625 – 632.

[29]Nichols, R., "A Summary of the Turbulence Models in the CREATE-AV Kestrel Flow Solvers," *2019 AIAA SciTech Forum*, 2019-1342.

[30]Bond, R., Tatum, K., Power, G., and Tuckey, T., "Capabilities of HPCMP CREATETM-AV Kestrel v11 for Hypersonic Flight and Ground Testing with a Two-Temperature Model," *2021 AIAA SciTech Forum*, 2021.

[31]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.

[32]Bond, R. and Stefanski, D., "Extension of Kestrel to General Thermochemical Models, Part II," *2018 Joint Thermophysics and Heat Transfer Conference*, 2018, pp. 2018–3267.

[33]Bond, R., Nichols, R., and Power, G., "Extension of Kestrel to General Thermochemical Models, Part I," *46th AIAA Thermophysics Conference*, Paper 2016-4435.

[34]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.

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

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

[37]Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43(2), 1981, pp. 357–372.

[38]Park, C., "Assessment of Two-Temperature Kinetic Model for Ionizing Air," *Journal of Thermophysics and Heat Transfer*, Vol. 3, No. 3, 1989, pp. 233–244.

[39]Gnoffo, P., Wood, W., Kleb, W., Alter, S., Glass, C., Padilla, J., Hammond, D., and White, J., "Functional Equivalence Acceptance Testing of FUN3D for Entry, Descent, and Landing Applications," *AIAA Computational Fluid Dynamics Conference*, 2013.

[40]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.

[41]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.

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

[43]Korzun, A., Nielsen, E., Walden, A., Jones, W., and Carlson, J., "Effects of Spatial Resolution on Retropropulsion Aerodynamics in an Atmospheric Environment," *2020 AIAA SciTech Forum*, Paper 2020-1749.

[44]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.

[45]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.

[46]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.

[47]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.

[48]Goodwin, G., Bachman, C., Johnson, R., and Kessler, D., "Synthetic freestream turbulence generation at an inflow boundary condition," *2021 AIAA SciTech Forum*.

[49]"LINK3D Mesh generation software," `https://www.link-3d.com`.

[50]Dong, G., Fan, B., and Ye, J., "Numerical investigation of ethylene flame bubble instability induced by shock waves," *Shock Waves*, Vol. 17, 2008, pp. 409–419.

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

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

[53]Nielsen, T., Edwards, J., Chelliah, H., Lieber, D., Goyne, C., Rockwell, R., and Cutler, A., "Hybrid LES/RANS simulation of supersonic premixed ethylene combustion," *2018 AIAA SciTech Forum*, Paper 2018-1145.