Implicit Thermochemical Nonequilibrium Flow Simulations on Unstructured Grids using GPUs

Gabriel Nastac, Aaron Walden, Eric Nielsen

NASA Langley Research Center

Hampton, Virginia, 23681

Abdelkader Frendi

University of Alabama in Huntsville

Huntsville, Alabama, 35899

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Motivation

- Current plans for U.S. exascale systems rely on GPU acceleration
  - 130 out of 500 fastest supercomputers (6 out of top 10) utilize GPU hardware
- Port to GPU architectures positions FUN3D, an unstructured-grid CFD solver, for this emerging landscape
  - Dramatically reduced run times enable early penetration of high-fidelity modeling
  - Ability to elucidate unprecedented physics – temporal, spatial, physical models
- The perfect gas path of FUN3D has been previously ported to NVIDIA Tesla GPUs
  - Ensembles of retropropulsion simulations using several billion elements have been performed on Summit, which debuted as the world's top system in 2018
  - Ensemble run-times reduced from ***years*** on a capacity-governed CPU system to a ***workweek*** on a leadership-class GPU architecture managed with a capability policy
- Here we port the generic gas path of FUN3D, which models thermochemical nonequilibrium flows including atmospheric entry, hypersonics, and combustion

Current HPC Landscape

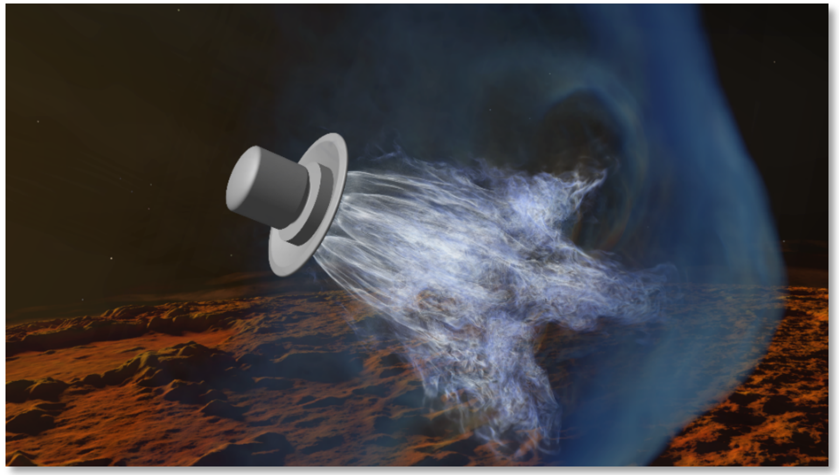
2. ORNL Summit (149 PF)
46. NASA Pleiades (6 PF)
53. NASA Electra (5 PF)
71. NASA Aitken 2 (4 PF)
168. NASA Aitken (2 PF)

New US Systems in 2021-2023

ANL Aurora (1000 PF)
ORNL Frontier (1500 PF)
LLNL El Capitan (2000 PF)

Architecture: CPU / GPU

PF: PetaFLOPS, or 10^{15} Floating-Point Operations Per Second



Governing Equations and Numerical Implementation

- Conservation of species, momentum, energies, and turbulence variables
- Two-temperature model available for thermal nonequilibrium
- Spalart-Allmaras turbulence model with Catris-Aupoix compressibility correction; DES option
- Variable species, energies, and turbulence equations
- Node-based finite-volume approach on general unstructured grids
- Fully implicit formulations are used to integrate the equations in time
    - Sparse block linear system: $A\boldsymbol{x} = \boldsymbol{b}$
    - Matrix A composed of diagonal and off-diagonal $N_{eq}\,x\,N_{eq}$ blocks
    - Memory and solution time increases as $O\left(N_{eq}^2\right)$
- System solved with multicolor point-implicit approach

$$\frac{\partial}{\partial t}(\rho y_s) + \frac{\partial}{\partial x_j}(\rho y_s u_j) - \frac{\partial}{\partial x_j}(J_{sj}) = \dot{\omega}_s$$

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_i u_j + p\delta_{ij}) - \frac{\partial}{\partial x_j}(\tau_{ij}) = 0$$

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_j}\left((\rho E + p)u_j\right) - \frac{\partial}{\partial x_j}\left(u_k\tau_{kj} + \dot{q}_j + \sum_{s=1}^{N_s} h_s J_{sj}\right) = 0$$

$$\frac{\partial}{\partial t}(\rho E_v) + \frac{\partial}{\partial x_j}(\rho E_v u_j) - \frac{\partial}{\partial x_j}\left(\dot{q_V}_j + \sum_{s=1}^{N_s} h_{V_s} J_{sj}\right) = S_v$$

$$\frac{\partial}{\partial t}(\rho\tilde{\nu}) + \frac{\partial}{\partial x_j}(\rho\tilde{\nu}u_j) - \frac{\partial}{\partial x_j}\left(\frac{1}{\sigma}\left(\mu\frac{\partial\tilde{\nu}}{\partial x_j} + \sqrt{\rho}\tilde{\nu}\frac{\partial\sqrt{\rho}\tilde{\nu}}{\partial x_j}\right)\right) = S_{\tilde{\nu}}$$

$$\boldsymbol{q} = [\rho\vec{y}_s, \rho\vec{u}, \rho E, \rho E_v, \rho\tilde{\nu}]^T$$

$$\int_{\boldsymbol{V}} \frac{\partial \boldsymbol{q}}{\partial t} dV + \oint_{\boldsymbol{S}} (\boldsymbol{F}\cdot\boldsymbol{n})dS - \int_{\boldsymbol{V}} \boldsymbol{S}dV = \boldsymbol{0}$$

$$\left[\frac{V}{\Delta\tau}\boldsymbol{I} + \frac{V}{\Delta t}\boldsymbol{I} + \frac{\partial\widehat{\boldsymbol{R}}}{\partial\boldsymbol{q}}\right]\Delta\mathbf{q} = -\mathbf{R}(\mathbf{q}^{n+1,m}) - \frac{V}{\Delta \mathrm{t}}(\mathbf{q}^{n+1,m} - \mathbf{q}^{\mathrm{n}})$$

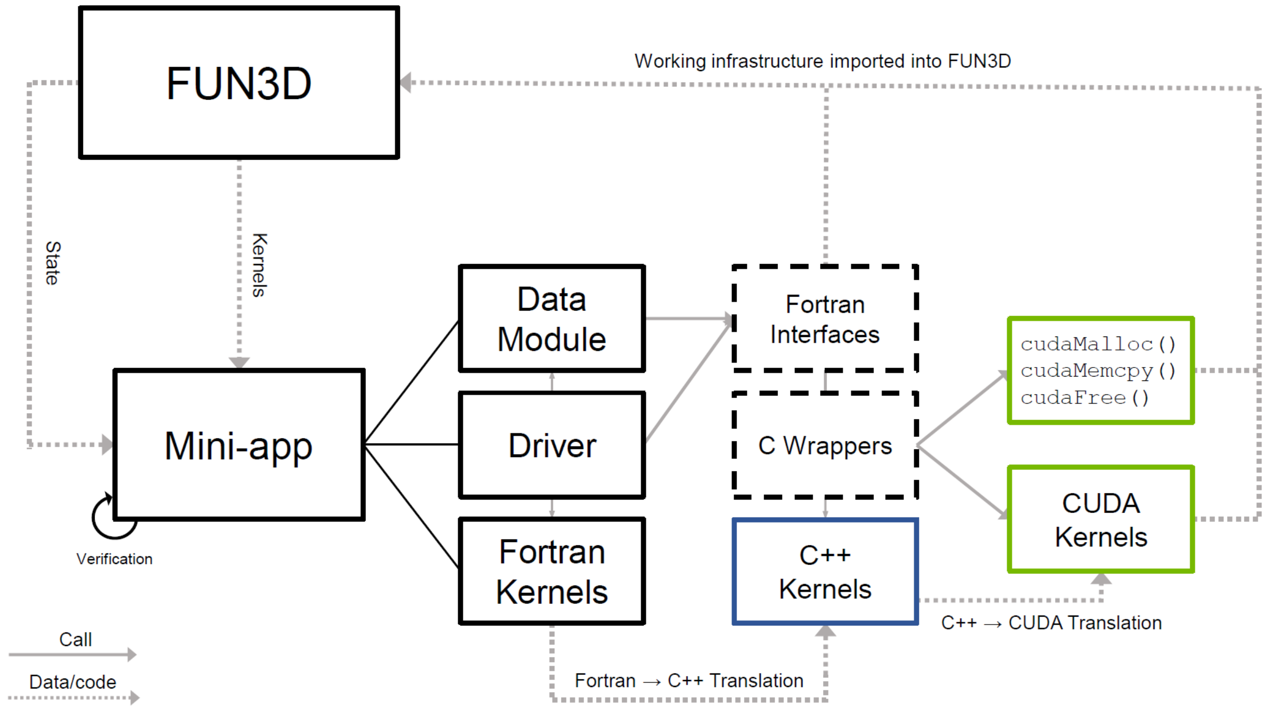
$$\mathbf{q}^{n+1,m} = \mathbf{q}^{n+1,m} + \Delta\boldsymbol{q}$$

GPU Design Approach

- Nomenclature: Host = CPU, Device = GPU
- FLUDA Library
  - CUDA C++ port of compute kernels in FUN3D
  - No external libraries are required
  - Effectively C++
  - Use of library in FUN3D is controlled by a run-time parameter
- Pre-processing routines remain on the host
- All PDE kernels performed on device
- Minimal data transfer between host/device (mainly scalars)
  - Large data motion at user-specified frequencies (e.g., restarts, visualization support)
- Data structures are identical between CUDA and Fortran contexts
  - Column-major order array layouts
  - GPU “mirror” data structures that match CPU data structures
  - Variable precision is identical to CPU approach
    - FP64 for most variables, with mixed FP32/FP64 for linear algebra

GPU Implementation

- Mini-app utilizes an entire state of FUN3D to perform a full iteration of the solve
- CPU and GPU kernels can be run at the same time and have outputs compared
- Once RMS norm of outputs is within specified tolerance (10^{-14} for FP64, 10^{-7} for FP32), kernels are integrated into FUN3D
    - Most kernels match to machine precision
    - Individual FP values generally do not match to machine precision due to order-of-operations; further complicated by asynchronous execution
    - Behavior not unique to GPUs; also observed on CPUs with random loop permutations



FUN3D mini-app structure and porting workflow

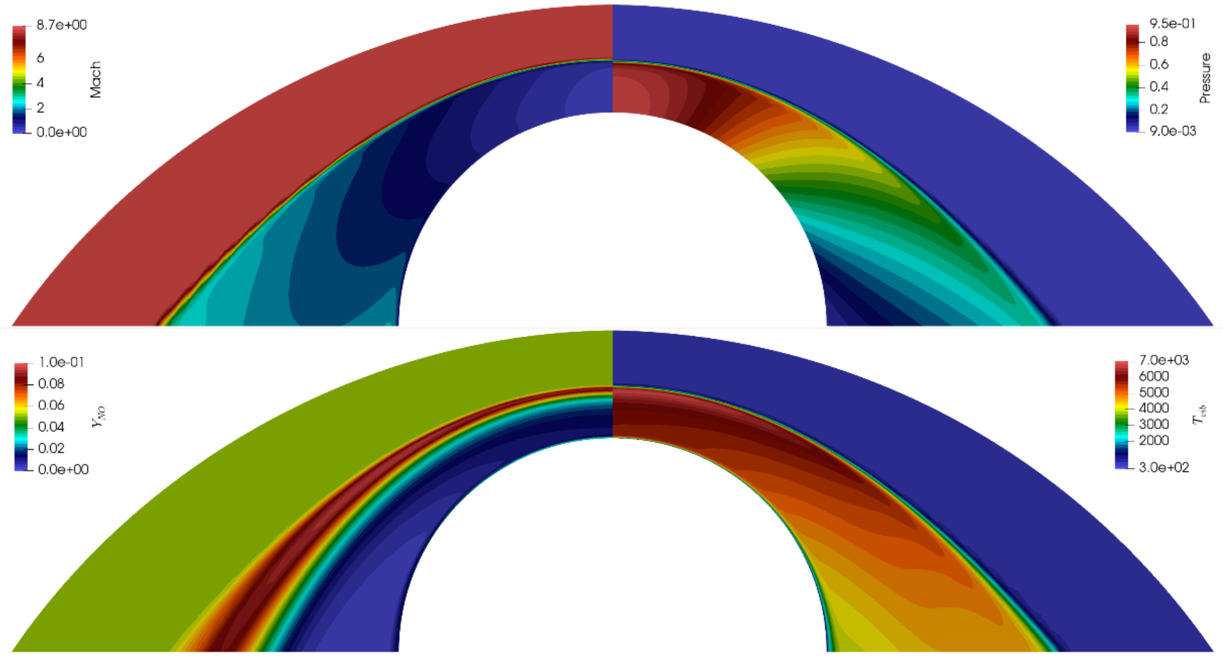
GPU Optimizations

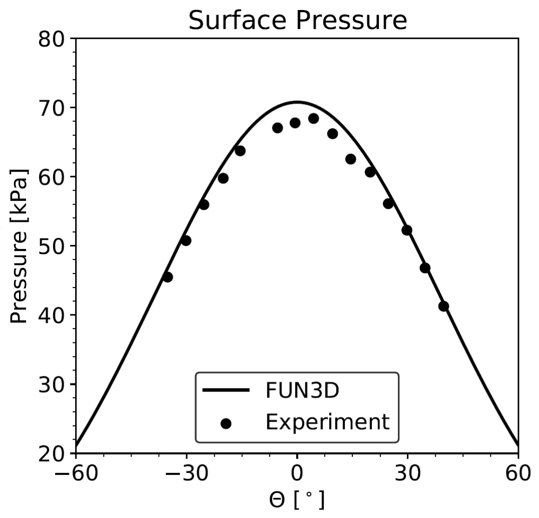
- Reduction of kernel state
  - Fortran implementation utilizes variable-length arrays (VLAs) for workspace
    - Since VLAs do not exist for C++, templating is extensively used
  - Initial naive CUDA port resulted in stack frames so large that the GPU ran out of memory immediately
  - To remedy this, multiple threads are assigned to a work item (such as a Jacobian) which reduces 2D arrays to scalars in many cases
  - Registers and shared memory are heavily used
- Reduce thread divergence
- Coalesced memory accesses
- Kernel launch parameter optimization
- See the paper for more details

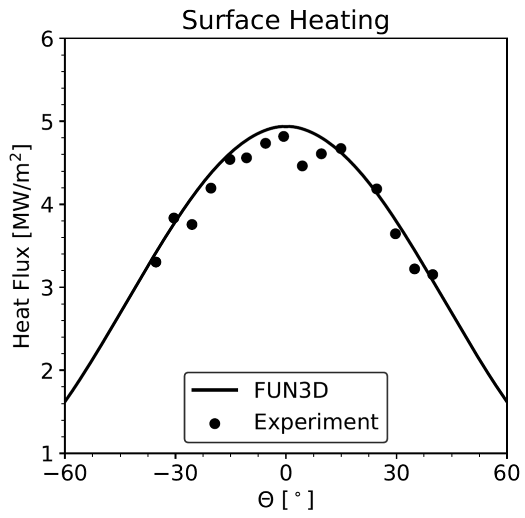
Mach 8.7 High Enthalpy Cylinder Flow

- Small verification case used during mini-app development
- Structured hexahedral grid: 65k~ nodes
- Five-species air with two-temperature model
- All equations converged to machine precision
- See the paper for details

	C_D
CPU	$4.3791550447125\times10^{-2}$
GPU	$4.37915504471\mathbf{01}\times10^{-2}$
Relative Difference	5.5×10^{-13}

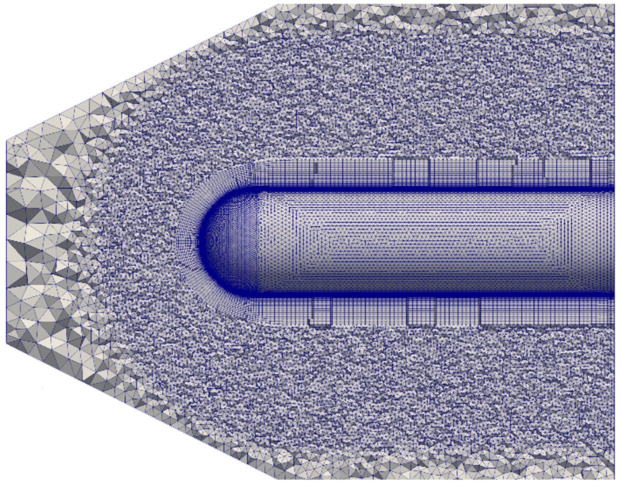




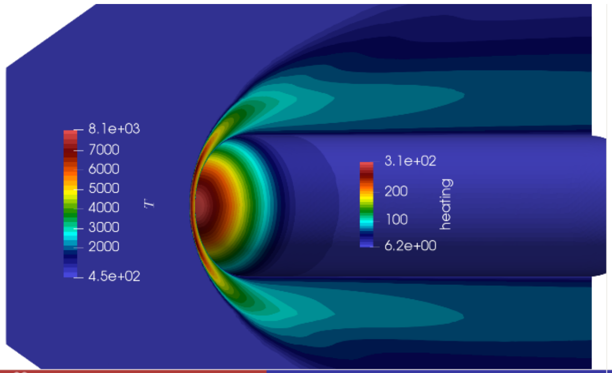


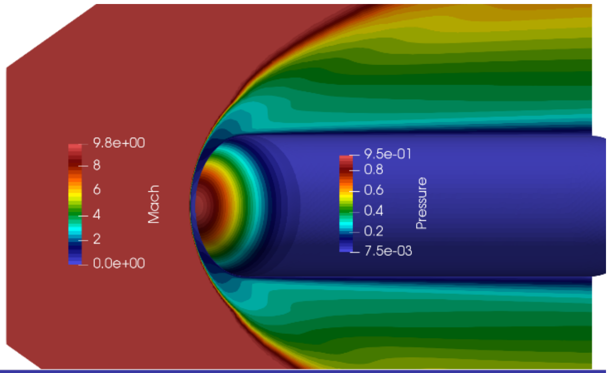
Mach 9.8 Hemisphere Cylinder Flow

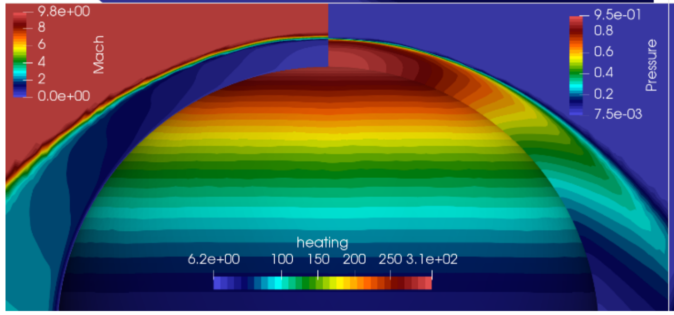
- Representative blunt body
- Unstructured mixed-element grid
  - 2.4M nodes
  - 6.5M tetrahedra, 2.6M prisms
- Five-species air with two-temperature model
- See the paper for details

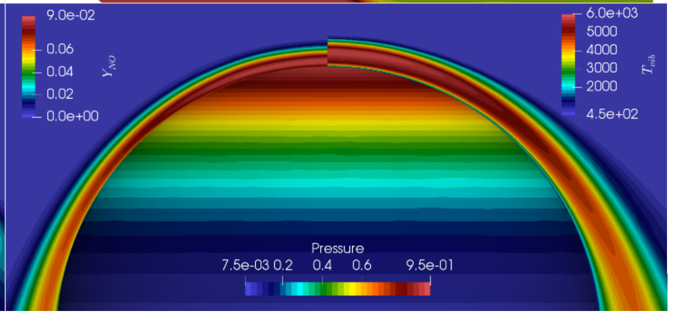


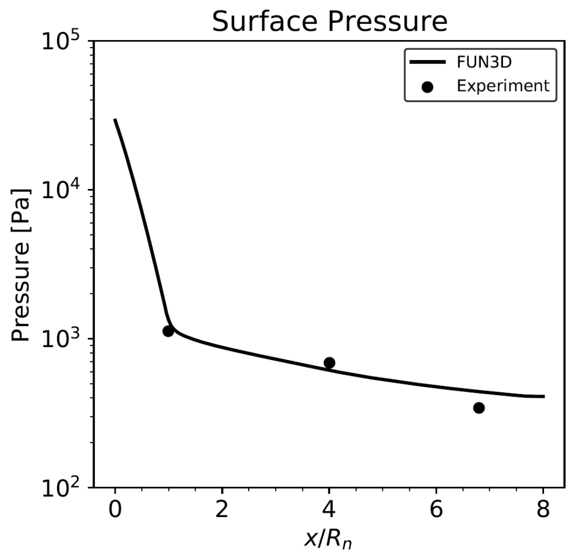
	C_D
CPU	1.973573461×10^{-3}
GPU	$1.97357\mathbf{1270}\times10^{-3}$
Relative Difference	1.110×10^{-6}

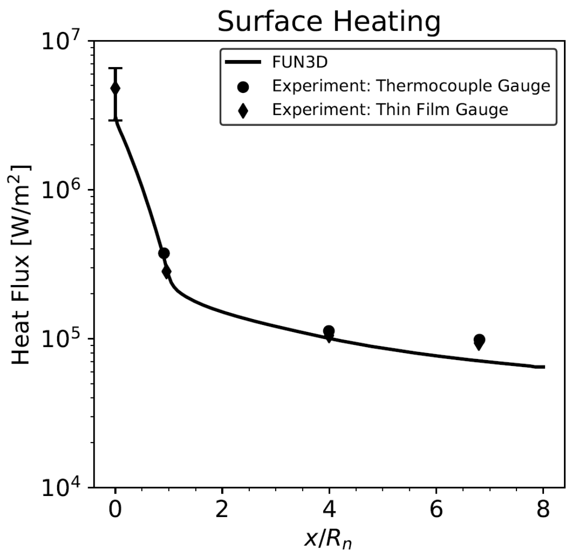






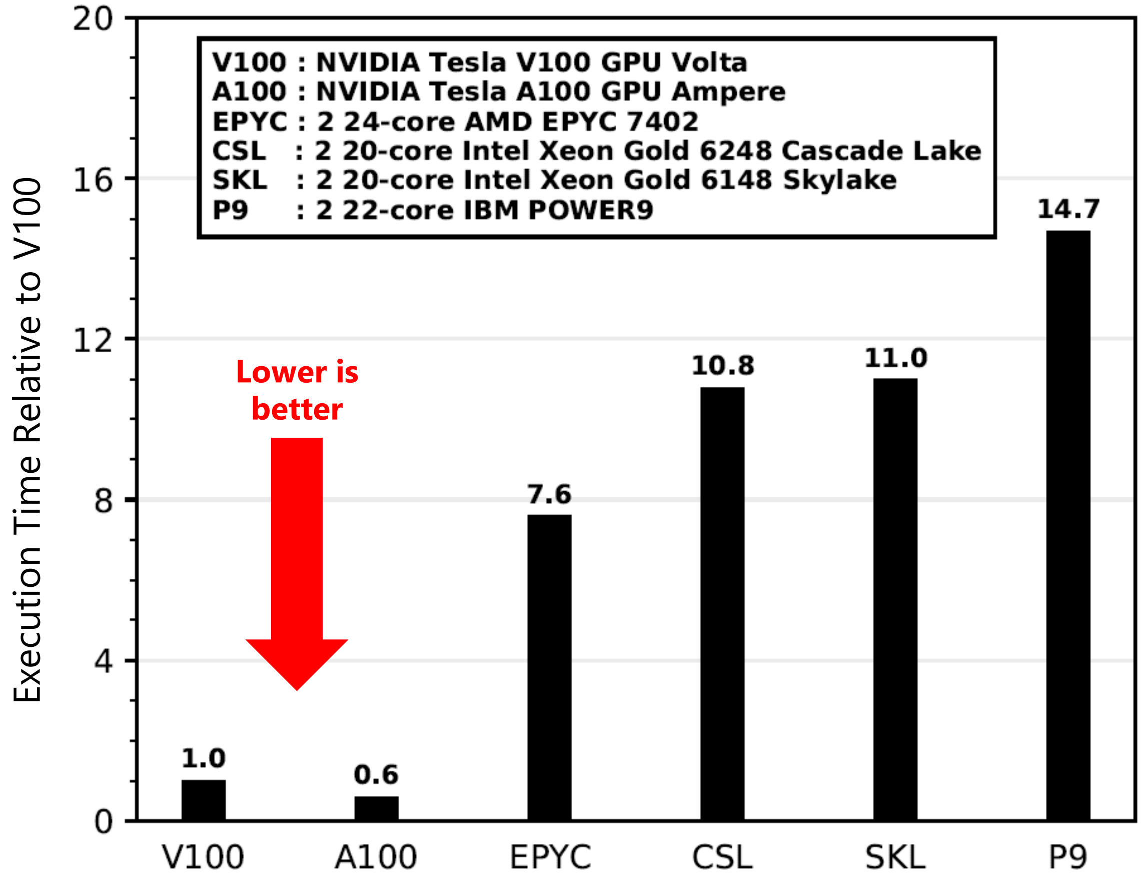






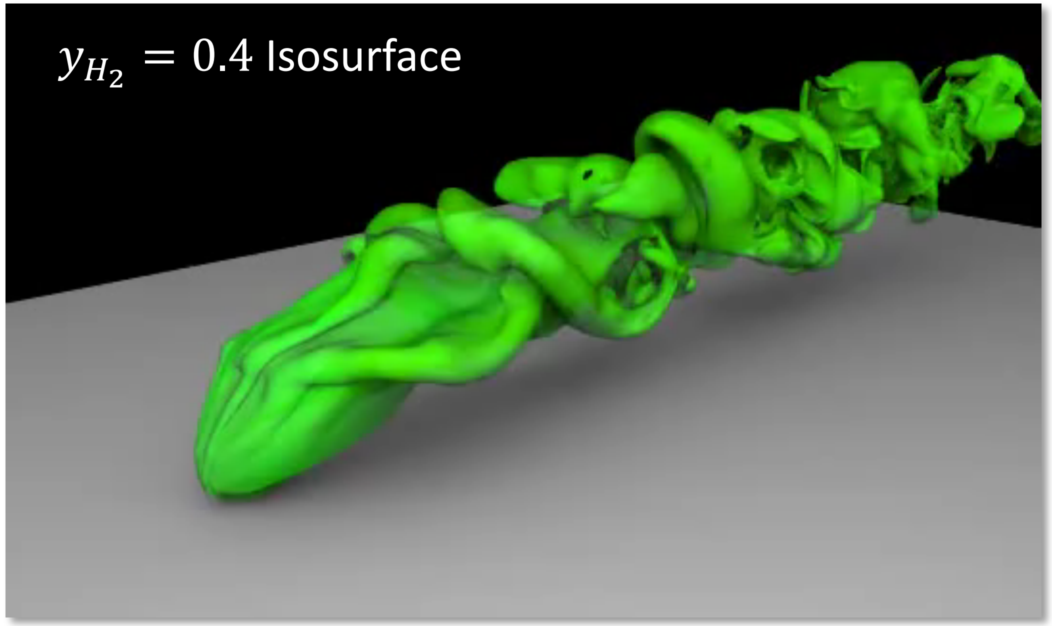
Mach 9.8 Hemisphere Cylinder Flow

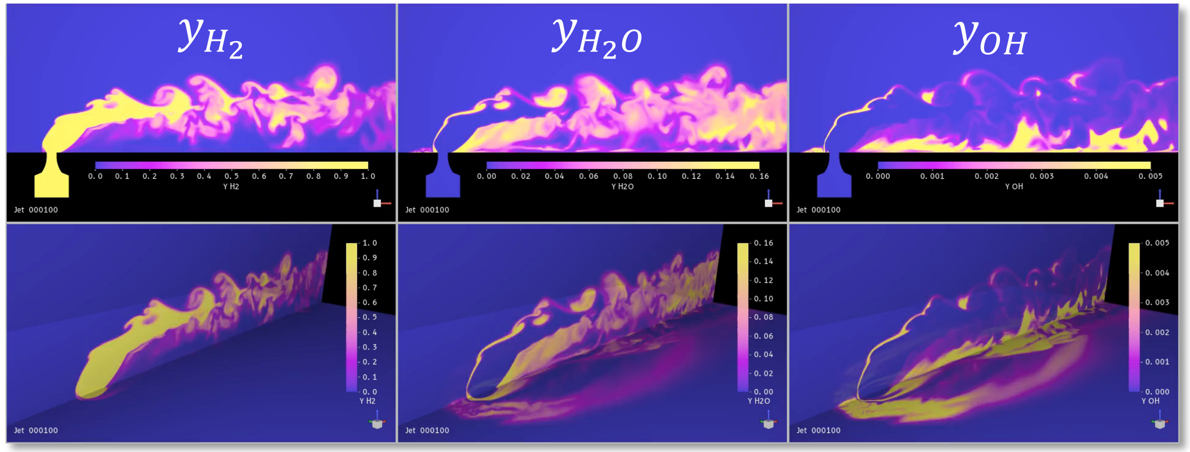
- Memory-bound applications should exhibit speedups commensurate with hardware memory bandwidth ratio
  - E.g., perfect gas FUN3D shows 4.5x speedup for NVIDIA Tesla V100 over dual-socket Intel Xeon
- Generic gas CPU implementation is not optimal
  - Templates are not natively available in Fortran
  - Optimizations (e.g., reduction of workspace, transpose) have not been performed on CPU
- Relative timings shown for nominal nonlinear step on a single device
  - Speedup reduced by 25% for steps with frozen Jacobian
  - Some minor model variations
  - Some variations due to grid topology
  - See paper for more details

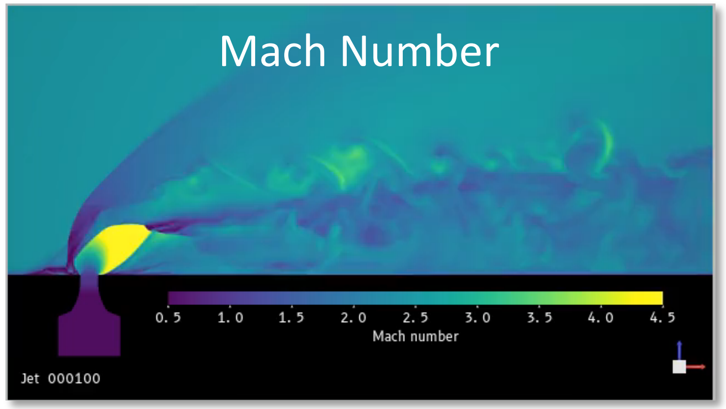


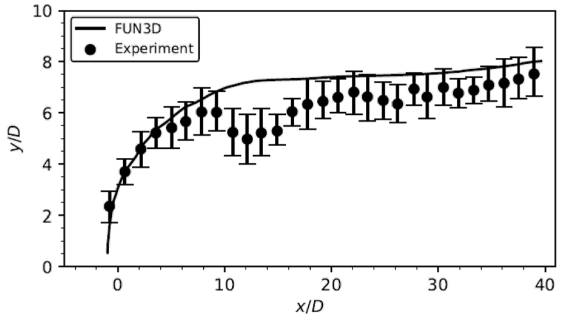
Transverse Hydrogen Jet in Supersonic Cross Flow

- Mach 2.4 cross flow and J=5 jet with pure hydrogen plenum
- 9-species hydrogen combustion mechanism, one-temperature model
- SA-Catris model with DES option: $N_{eq} = 14$
- 57M nodes, 278M tetrahedra, 22M prisms
- BDF2 time integration with 5 subiterations
- Flow Through Time (FTT):
  - 48 NVIDIA Tesla V100s: 9.3 hours
  - 4000 Intel Xeon Skylake cores: 43.0 hours
  - Est. 20,000 Intel Xeon Skylake cores required to keep pace
    - ➢ 1 V100 ≈ 417 Skylake cores at this scale
- See paper for more details





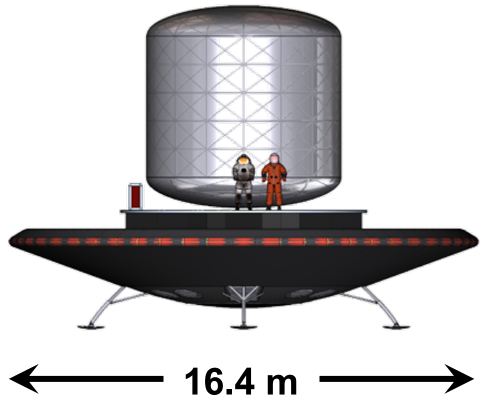


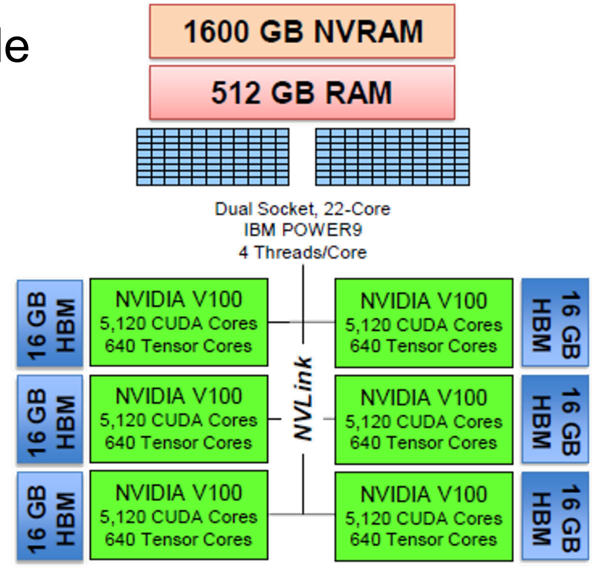


Jet height versus axial distance

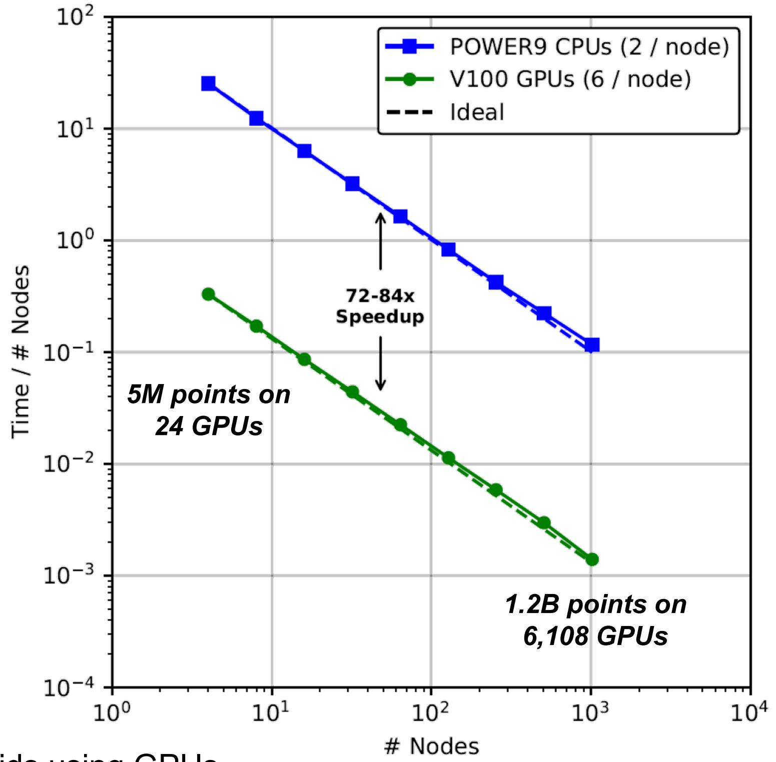
Performance at Scale

- Weak scaling evaluated on Oak Ridge Summit system
- Node consists of 2 22-core IBM POWER9 CPUs, 6 NVIDIA Tesla V100 GPUs
- Mars retropropulsion unstructured grids used
  - 10 species, one-temperature model, SA-DES ($N_{eq} = 15$)
  - Mars atmospheric composition
  - 8 Plena compositions are products of methane-oxygen combustion
- Each run places 1.2M grid points/node, or 200K grid points/GPU
- CPU- and GPU-only executions scale linearly
- GPUs retain ~75x node-level speedup at scale
- 1.2 billion points on 1,018 nodes
  - One physical time step with BDF2 takes about one second
  - Performance equivalent to several million CPU cores





Summit Node



Summary

- Generic gas path of FUN3D has been successfully ported, optimized, and verified for NVIDIA Tesla GPUs
- One NVIDIA Tesla V100 equivalent to ~400 Intel Xeon Skylake cores at scale
- Benchmarks have been performed using over 6,000 GPUs with grids containing several billion elements
  - Performance equivalent to several million CPU cores

Future Work

- Decoupled approaches to reduce memory requirement and increase performance
- Lower dissipation numerics
- Mixed-precision arithmetic
- Nonlinear solver improvements