

Athena-RFX

Athena-RFX is a high-fidelity, computational fluid dynamics (CFD) simulation tool designed to model complex turbulent reacting flow processes using direct numerical simulations (DNS).

Key features include:

- Coupling of stiff, finite-rate chemical kinetics using 2nd-order accurate temporal operator-splitting with YASS ODE solver [1] for reaction kinetics of varying complexity.
- Support for arbitrary equation-of-state (EOS) models to capture variety of ideal-gas and more realistic reacting mixtures – from chemical to thermonuclear.
- Support for diffusion and conduction in multi-component chemical mixtures.
- Support for gravitational potentials and magnetic fields for astrophysical and magneto-hydrodynamic simulations.
- Proven distributed memory parallel scalability to ~100k cores.
- High-accuracy, upwind hydrodynamic solver for fluid advection and transport.

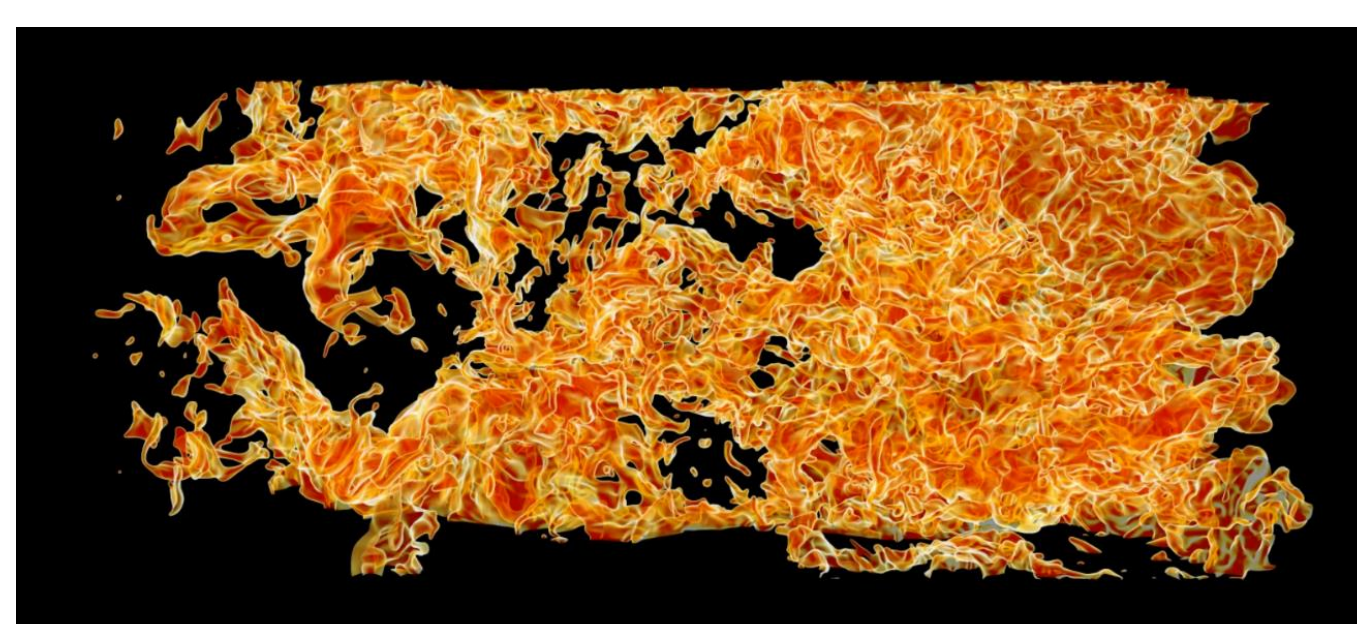
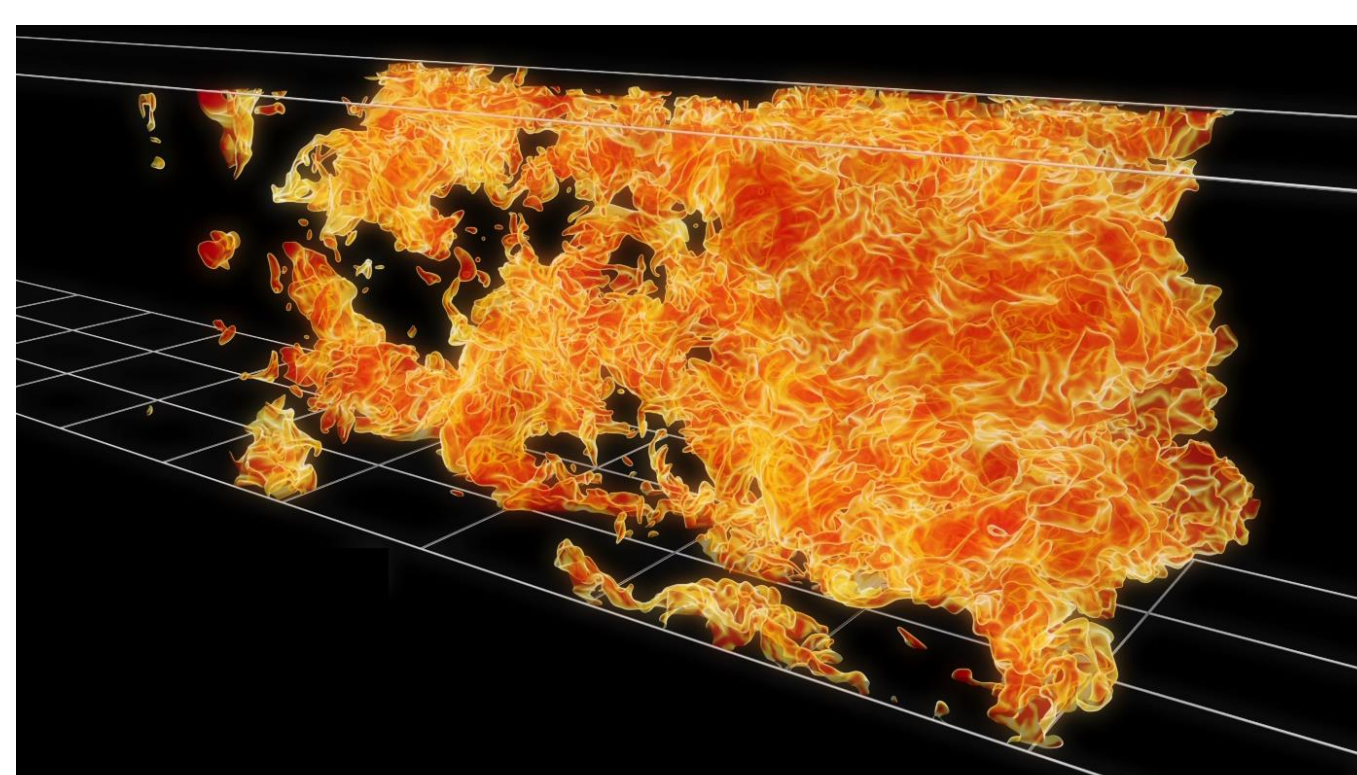


Figure 1. Turbulent methane-air flames simulated with Athena-RFX multi-physics models. Images courtesy of the DoD HPCMP Data Analysis and Assessment Center (DAAC).

YASS ODE Solver Vectorization

The N_k chemical species (fuel, oxygen, etc.) are changed by chemical reactions each CFD timestep after turbulent advection and diffusion.

- Each cell is independent and modeled as a set of $(N_k + 1)$ ordinary differential equations (ODEs) that must be integrated in time over the CFD timestep (δt).
- YASS integration can be the dominant cost for large chemical reaction mechanisms (e.g., ~60% run-time with 50 chemical species).
- ODE system integrated with YASS stiff solver.
 - Semi-implicit, single-step method avoids need to solve nonlinear system.
 - ODE step size (h) adapted to control truncation errors and solution stability. Candidate solutions can be rejected if stability not ensured.
 - ODE right-hand-side vector and dense Jacobian matrix formed using tabulated reaction rates and analytical mass-action relations defined by arbitrary chemical kinetics mechanisms.

SIMD Challenges:

- Linearly-interpolated reaction rates:
 - Small, uniform spacing ($\delta T = 1K$) leads to a large table (i.e., cache pressure) and higher rate of diverging SIMD lanes.
- Packed dense iteration matrices:
 - Exceed L1 cache for larger kinetics models
 - Standard LU factorization libraries do not support packed SIMD format with partial row pivoting.
 - Row pivoting serializes when pivot row is different across lanes.
- CFD cells in or near the flame region may require more iterations than non-reacting cells leading to SIMD load imbalance and reduced efficiency.

Scalar Algorithm:

```
forall c in Cells:
  y_k ← getChemicalComposition(c)
  T ← getTemperatureEOS(y_k, getEnergy(c))
  t ← 0; h ← setInitialStepsize(c)
  while t < t_end:
    k_r ← getTabulatedArrheniusRates(T)
    [y_k, T] ← getNetRatesAndJacobian(y_k, T, k_r)
    [Δy_k, ΔT] ← solveLinearSystem(J, h[y_k, T])
    err ← getEstimatedError([Δy_k, ΔT])
    if err < ε:
      [y_k, T] ← [y_k, T] + [Δy_k, ΔT]
      t ← t + h
    h ← setStepsize(err)
  c ← setChemicalStateChanges(y_k, T)
```

SIMD* Algorithm:

```
forall c in Cells by Stride V:
  y_k ← getChemicalComposition(c)
  T ← getTemperatureEOS(y_k, getEnergy(c))
  t ← 0; h ← setInitialStepsize(c)
  NotFinished ← True
  while Any(NotFinished):
    k_r ← getTabulatedArrheniusRates(T)
    [y_k, T] ← getNetRatesAndJacobian(y_k, T, k_r)
    [Δy_k, ΔT] ← solveLinearSystem(J, h[y_k, T])
    err ← getEstimatedError([Δy_k, ΔT])
    accept ← NotFinished ∧ (err < ε)
    [y_k, T] ← Where(accept, [y_k, T] + [Δy_k, ΔT])
    t ← Where(accept, t + h)
    h ← setStepsize(err)
    NotFinished ← t < t_end
  c ← setChemicalStateChanges(y_k, T)
```

*Vector elements are in bold. Vector length (V) is arbitrary but commonly 2ⁿ (e.g., 4 or 8).

Goals and Strategy

The focus of this project is to investigate the potential performance improvement of multi-physics models by exploiting single-instruction, multiple-data (SIMD) parallelism on modern HPC platforms.

- SIMD parallelism on modern HPC CPU (host) platforms has quadrupled in the past decade and accounts for substantial percentage of the total FLOP rate.
 - 128-bit (SSE4, c. 2011) to 512-bit (AVX-512, c. 2016)
 - Without efficient vectorization, applications are limited to approximately 1/8th of the available FLOPs on modern HPC systems.
- Multi-physics models such as chemical kinetics and diffusion rates can be evaluated concurrently in each CFD mesh cell exposing substantial parallelism. Exploiting SIMD parallelism is complicated by their complexity and algorithmic structure.
 - Most models are designed to solve one ‘point-problem’ per evaluation and must be refactored to solve multiple cells concurrently (e.g., 4 or 8 at once).
 - Many models solve nonlinear problems using iterative methods which can be inefficient in SIMD environment: must iterate all problems until the last converges.
- Ideally, compilers will identify and vectorize loops automatically. But, automatic compiler vectorization is inefficient or inadequate for most multi-physics models of interest. Instead, we explore two techniques for *explicit* vectorization to expose the fine-grain SIMD parallelism.
 - C++ SIMD datatype templates (VCL, *vector class library* [2]) to overload original (C99) scalar implementations with minor modifications. SIMD vector operators map to SIMD compiler intrinsics.
 - Fortran fixed-length arrays (e.g., 4, 8, 16) combined with whole-array expressions and alignment directives. Whole-array expressions simplify refactoring and mimic datatype overloading. This still relies on a vectorizing compiler to replace fixed-length loops with suitable vector operations based on analysis or explicit directives.
 - Modify multi-physics function interfaces and call sites to send a block of cells instead of one per call. Often requires array-of-structures (AOS) (i.e., 1 object per cell) to structure-of-arrays (SOA) repacking.

Vectorized YASS ODE Solver Results

Evaluated performance of YASS solver within Athena-RFX on HPC systems with AVX2 or AVX-512 SIMD support.

- Simulating Hydrogen-Air and Ethane-Air “premixed” flames with 2²⁰ mesh cells.
- Benchmark Specification:
 - Onyx: Cray XC-50 system at US Army Engineering Research & Development Center (ERDC), Intel Xeon “Broadwell” (BRD) E5-2699v4 and Intel Xeon Phi “Knights Landing” 7230 (KNL).
 - Stampede2: Dell/Intel system at the Texas Advanced Computing Center (TACC), Intel Xeon “Skylake” Platinum 8160 (SKX) CPUs.
 - Using one compute node. Timings are averaged for last 20 timesteps to avoid start-up interference.
 - Workloads: 32 MPI processes per node on BRD and SKX (32k cells per core) nodes; 64 MPI processes per node on KNL (16k cells per core) nodes.
 - SIMD vector width is 4 on BRD and 8 on KNL and SKX matching the native AVX2 and AVX-512 size with double-precision (64-bit) words.
 - Compilers: Intel (v17) and Cray (v8.6) C/C++/Fortran.
 - Smaller Hydrogen-Air reaction mechanism is based upon the full UCSD [4] reaction mechanism.
- Analysis:
 - KNL shows higher relative speed-up than SKX with the same vector length. Scalar run-time on SKX ~4x faster than KNL lowering the net parallel efficiency to ~50%.
 - Smaller H₂-Air mechanism gives proportionally greater speed-up than the full mechanism, likely due to lower L1 data and instructional cache pressure.
 - Fortran vectorization matches or exceeds VCL approach on both large and small mechanisms with Intel compiler and has superior cross-compiler performance portability. This is counter to the EOS findings (Fig. 2).
 - Aggressive function in-lining (IPO) has minimal or even negative impact on SIMD performance though it is generally useful for scalar implementations.

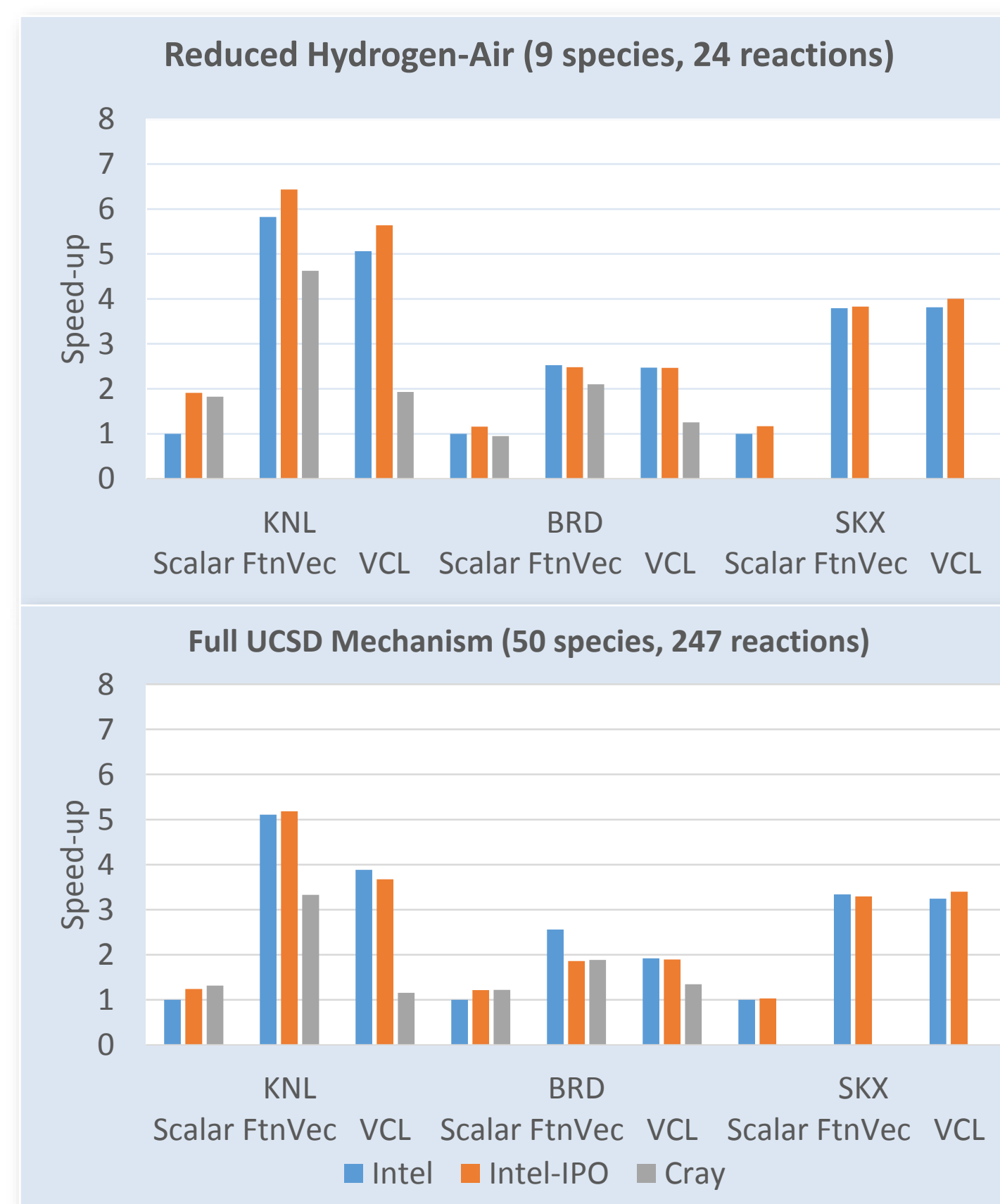


Figure 3. Speed-up relative to scalar baseline of the YASS integrator using Intel compiler suite (v17) with small (hydrogen-air) and large (ethane-air) chemical kinetics mechanism [4] on Intel Xeon Phi (KNL), Broadwell (BRD), and Skylake (SKX). Fortran and VCL vector lengths match native size (i.e., 4 on BRD and 8 on KNL & SKX). Comparing Intel compiler with and without inter-procedural optimization (IPO) and using the Cray Fortran/C/C++ compiler suite, where available. KNL and BRD results from Onyx (ERDC); SKX results from Stampede2 (TACC).

Multicomponent Equation-of-State (EOS) Vectorization

Many multi-physics models require the gas temperature (T) as input. In Athena-RFX, T is not explicitly known and must be evaluated from the thermal energy (e) and chemical concentrations (y_k) with a multi-component, ideal-gas equation-of-state (EOS).

- The EOS is nonlinear and is iteratively solved using Newton’s method.
- Tabulated 5th-order polynomials [3] relate T and e within specific temperature ranges for each chemical component.
 - Two sets of coefficients for T above and below 1000k.
- Coefficient table lookup and iterative convergence must be handled explicitly with SIMD parallelism.
 - Replace single convergence test with collective SIMD vector test and masked T update.
 - If lanes have T ’s that span the 1000k tabulation divide, must either evaluate both high and low T terms and conditionally select result or use parallel gather of the tabulated coefficients. (The later approach is shown here.)

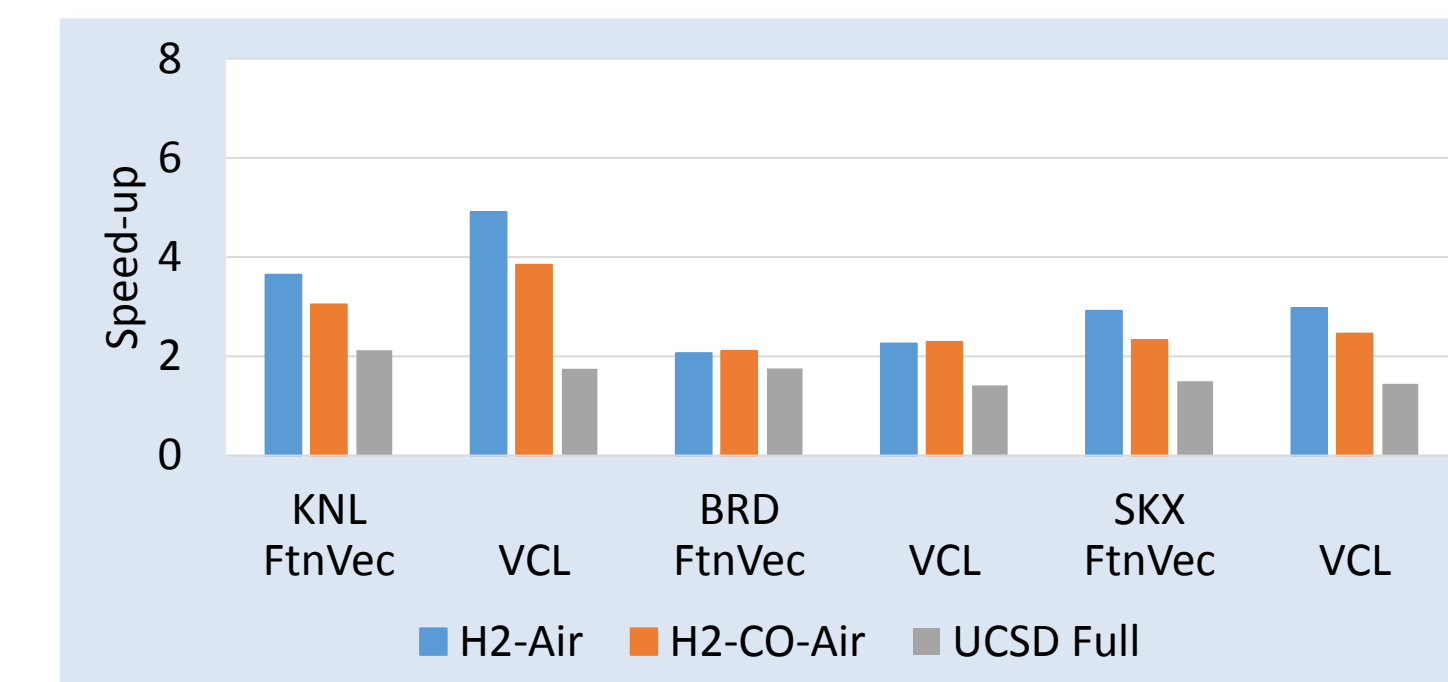


Figure 2. Speed-up relative to scalar baseline of the temperature evaluation via the multi-component, ideal EOS with three chemical mechanisms [4] (with 9, 14, and 50 chemical species) using Fortran (FtnVec) and C++ VCL vectorization techniques and the Intel C/C++/Fortran compiler. See Fig. 3 for more details.

Scalar Algorithm:

```
T_0 = T_guess
do
  forall k: a_{k,i} ← LookupCoefficients(T, k)
  F = e - Σ_{k=1}^{N_k} y_k [a_{k,6} + a_{k,1}T + a_{k,2}T^2 + a_{k,3}T^3 + a_{k,4}T^4 + a_{k,5}T^5]
  F' = Σ_{k=1}^{N_k} y_k [a_{k,1} + a_{k,2}T + a_{k,3}T^2 + a_{k,4}T^3 + a_{k,5}T^4]
  T_{n+1} = T_n - F/F'
while |F(e, y_k, T_{n+1})| > ε
```

SIMD* Algorithm:

```
T_0 = T_guess; NotConverged ← True
while Any(NotConverged)
  forall k: a_{k,i} ← Gather( LookupCoefficients(T, k) )
  F = e - Σ_{k=1}^{N_k} y_k [a_{k,6} + a_{k,1}T + a_{k,2}T^2 + a_{k,3}T^3 + a_{k,4}T^4 + a_{k,5}T^5]
  F' = Σ_{k=1}^{N_k} y_k [a_{k,1} + a_{k,2}T + a_{k,3}T^2 + a_{k,4}T^3 + a_{k,5}T^4]
  T_{n+1} = Where(NotConverged, T_n - F/F')
  NotConverged ← |F(e, y_k, T_{n+1})| > ε
```

*Vector elements are in bold

Summary and Future Directions

- Explicit SIMD refactoring shows potential for 5-6x speed-up on KNL and 3-4x on SKX on the costly chemical kinetics multi-physics models.
 - Net overall simulation speed-up of 2x possible for large kinetics models.
- Temperature evaluation via EOS demonstrates > 50% SIMD parallel efficiency on KNL in solving a nonlinear problem via an iterative scheme.
 - Similar EOS calculations have similar structure and algorithm and same approach can be replicated.
 - Nonlinear or iterative solutions come across many disciplines and can follow this approach.
- C++ datatype overload provides finer control and better speed-up for lighter weight, data-intensive kernels (e.g., EOS) but less performant than Fortran fixed-length array method for higher cost kinetics kernels.
 - Explicit VCL C++ aligned/unaligned load/store operators improve data throughput when loading data from CFD cell AOS data structures. Compiler alignment directives in fixed-length vector approach reduce or eliminate the peel loop overhead but cannot match the explicit load/store approach.
 - Fortran approach allows a compiler to perform thorough analysis and reorder & fuse operations to improve performance compared to explicit VCL implementation when load/store is a negligible cost.
- Future Directions:
 - Extend multi-physics SIMD refactoring to chemical diffusion and thermal conduction coefficient evaluation. Exploit experience optimizing parallel lookup of tabulated polynomial coefficients.
 - Explore higher-order interpolation within reaction rate tabulation to reduce table size, cache pressure and lane divergence.

Acknowledgements

This material is based upon work supported by, or in part by, the Department of Defense High Performance Computing Modernization Program (HPCMP) under User Productivity, Technology Transfer, and Training (PETTT) contract # GS04T09DB00017. Any opinions, finding and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DoD HPCMP.

References

- 1) Khokhlov, A., “YASS—yet another stiff solver that works,” NRL Technical Report (1998).
- 2) <http://www.agner.org/optimize/#vectorclass>
- 3) Gordon, S. and McBride, B.J., “Computer Program for Calculation of Complex Chemical Equilibrium Compositions, Rocket Performance, Incident and Reflected Shocks and Chapman-Jouguet Detonations,” NASA Report SP-273 (1971).
- 4) UCSD mechanism (2014) <http://web.eng.ucsd.edu/mae/groups/combustion/mechanism.html>