

Refactoring and Optimizing Multiphysics Combustion Models for Data Parallelism

Dr. Christopher P. Stone
US DoD HPCMP PETTT Program
(Contractor)
Athens, GA, USA
chris.stone.ctr@engility.com

Prof. Alexei Y. Poludnenko
Dept. of Aerospace Engineering
Texas A&M Univ.
College Station, Texas, USA
apoludn@tamu.edu

Dr. Brian D. Taylor
US Air Force Research Lab
Eglin AFB, Florida, USA
brian.taylor.56@us.af.mil

Categories and Subject Descriptors

D.1.3 Concurrent Programming.

General Terms

Performance, Algorithms.

Keywords

SIMD, Data Parallelism, Vectorization, Multiphysics.

1. INTRODUCTION

High-fidelity combustion simulations combine high-resolution computational fluid dynamics (CFD) numerical modeling methods with multi-physics models governing processes such as chemical kinetics and molecular transport (e.g., diffusion in multi-component mixtures). These multi-physics models can dominate the computational cost of a simulation, especially as the number of chemical species increases. Due to the high cost of combustion simulations and the important role they play in propulsion and power research, methods are needed to reduce the computational time and cost.

Multi-physics models tend to exhibit significant parallelism, especially in time-marching schemes based upon operator splitting (e.g., Strang [1] splitting). In this approach, multi-physics models within each mesh cell can be solved independently and concurrently using various parallel processing methods.

Single-instruction, multiple data (SIMD) vector parallelism is becoming more significant on recent HPC platforms in response to stagnant clock speeds. In SIMD parallelism, the same instruction is applied to multiple, independent data elements concurrently. SIMD capabilities have widened quickly, quadrupling over the past decade, with the latest, AVX-512, supporting 512-bit super-words.

The most common application of SIMD processing is loop-level parallelism where multiple loop iterations are computed concurrently. Vectorizing compilers can identify suitable loop structures and generate parallel code. However, they must be conservative to ensure correctness and will not necessarily vectorize all loops thus lowering the overall potential speed-up. Other factors reducing the effectiveness of automatic vectorization are data alignment and stride, nested loops, and complex conditional branching.

As noted, the same multi-physics model computations may be applied to each cell concurrently in the combustion applications of interest here. However, the complex algorithmic structure of the multi-physics models and their iterative nature pose significant difficulties for automatic vectorization. Our approach is instead to

explicitly vectorize the models so that multiple cells are evaluated concurrently. We have evaluated operator-overloading methods using C++ SIMD vector datatypes (i.e., the VCL C++ vector class library [2]) as well as Fortran whole-array expressions combined with fixed-length arrays. Both approaches are evaluated in terms of performance across different vector widths as well as platforms and compilers. We have applied these methods to two important models: stiff integration of finite-rate chemical kinetics and gas equation-of-state (EOS) evaluation in multi-component mixtures.

In operator-split combustion simulations, the finite-rate kinetics in each cell requires the integration of a set of coupled, stiff ordinary differential equations. Each cell can be solved concurrently. At the same time, key features such as the iterative nature of the stiff integration solver, table lookup and interpolation of the reaction rates, and stable matrix inversion of multiple linear systems (e.g., row pivoting) are difficult to implement efficiently in the SIMD data parallel context.

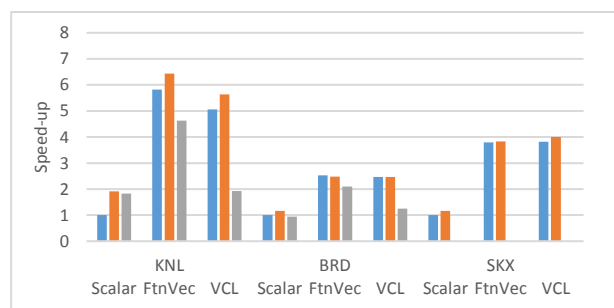
Many multi-physics models require the gas temperature (T) as input. In the current modeling context, T is not directly evolved on the grid and must instead be evaluated from the (conserved) sensible energy (e) and chemical composition using a multi-component, ideal-gas EOS. The EOS is nonlinear with respect to T and is iteratively solved using Newton's method. Tabulated 5th-order polynomials [3] relate T and e within specific temperature ranges¹ for each chemical component. As with kinetics, table lookup of the polynomial coefficients and iterative root-finding must be handled explicitly with SIMD parallelism.

We briefly summarize the performance attained from the vectorization efforts for the finite-rate kinetics stiff integration and multi-component gas EOS evaluation. The final poster shall include expanded benchmark results as well as the implementation details (e.g., pseudo-code) to guide other developers and performance engineers in SIMD refactoring efforts. These will include methods for parallel table lookup with diverging SIMD lanes (e.g., lanes requiring different or the same table rows), data alignment control methods, and strategies for handling conditional computations while preserving vector parallelism.

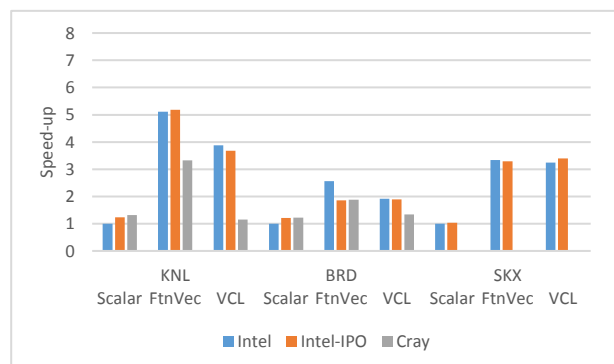
2. RESULTS

We evaluated the performance of the finite-rate kinetics and temperature calculation on three platforms with different vector widths (256 and 512 bit), instruction sets (AVX2, AVX-512), and compiler options. The performance using three different chemical

¹ For chemical species of interest here, two sets of polynomial coefficients are tabulated for temperatures above / below 1000K.



a) Hydrogen-air (9 Species, 24 Reactions)



b) Ethane-air (full UCSD mechanism) (50 Species, 247 Reactions)

Fig 1. Speed-up using VCL C++ templates and Fortran fixed-length array vectorization (FtnVec) methods on the (a) small and (b) large chemical kinetics mechanisms on Intel Xeon Phi Knights Landing (KNL), Broadwell (BRD), and Skylake Platinum (SKX) platforms.

reaction mechanisms with increasing numbers of species (9, 14, and 50) and reactions (24, 34, and 247) has been assessed.

The stiff reaction integration speed-ups relative to the baseline implementation (i.e., not explicitly vectorized²) with the small (hydrogen-air [4]) and large (ethane-air) mechanisms are shown in Figure 1. Fig 2 shows the SIMD parallel speed-ups for all three chemical mechanisms for the temperature evaluations via the nonlinear EOS using the Intel compiler.

The Intel Xeon Broadwell E5-2699v4 (BRD) (i.e., AVX2) and Xeon Phi 7230 Knights Landing (KNL) benchmarks were obtained on the Onyx Cray XC-50 system at the DoD Engineering Research and Development Center (ERDC). The Skylake Platinum 8160 (SKX) results were obtained on the *Stampede2* system at the Texas Advanced Computing Center (TACC). The BRD and SKX results are based on 32 MPI processes on one dual-socket node with 32k cells per process and the KNL results used 64 MPI processes on one device with 16k cells per process. All results are averaged over the last 20 timesteps to mitigate any program start-up interference (e.g., input file caching). The timings include compilation with and without inter-procedural optimization (IPO) which attempts to inline functions across the entire source code using the Intel C/C++/Fortran compiler suite. The Cray compiler is also shown for comparison when available. All kinetics speed-ups are relative to the Intel compiler suite without IPO.

The explicit vectorization width was set to match the native vector width on the platforms using double precision arithmetic. That is, each kinetics integration or EOS evaluation solved eight cells concurrently on the KNL and SKX platforms and four on the

² Note that this does not preclude compiler auto-vectorization of loops within the models as applied to a single mesh cell.

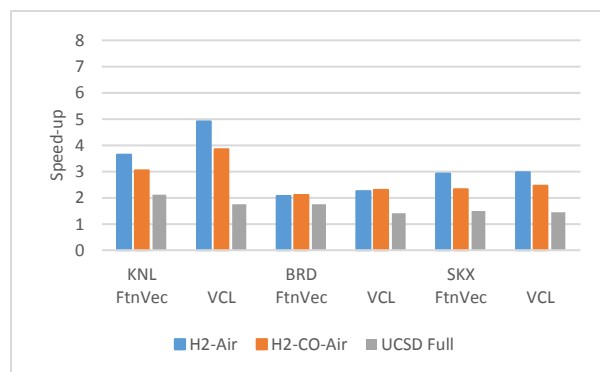


Fig 2. Speed-up relative to the scalar baseline of the temperature evaluation via the multi-component, ideal gas equation-of-state (EOS) in the three chemical mechanisms using Intel compiler suite.

BRD platform. Since the vector widths match on the KNL and SKX platforms, we expect similar parallel speed-up. However, we observe lower speedup on the SKX. This appears to be more dependent upon the increased scalar performance on the SKX than a reduction in vector performance. For example, the scalar run-time is 3.9x faster on SKX than KNL and only 2.4x faster with explicit vectorization with the medium sized, H₂-CO-Air, mechanism. Note that the scalar speed-up is higher than the ratio of the reported core clock speeds on the two platforms. This effect is observed in both the kinetics integration and the EOS evaluation kernels.

The implementation techniques (i.e., fixed-length arrays with compiler directives or VCL C++ templates) do not consistently impact the kinetics integration performance but do consistently improve the EOS evaluation performance. Both implementations included strategies to control the data alignment for efficient vector loads/stores and conditional (i.e., masked) computations. However, VCL C++ member methods provide finer control over load/stores and conditional (masked) computation that are beneficial for the EOS kernel due to its lower computational intensity compared to the kinetics kernel.

The performance improvements due to the explicit vectorization decrease as the number of chemical species increase (i.e., the chemical mechanism size increases). Loops with range approximately equal to the number of species exist in the scalar implementations and can be automatically vectorized reducing the overall potential speed-up from explicit vectorization. This is most pronounced in the EOS kernel since the majority of operations³ can be expressed as parallel loop iterations.

Overall, we observed a significant performance improvement using both explicit vectorization implementation approaches. These speed-ups have a tangible impact on the total simulation run-time and computational efficiencies.

ACKNOWLEDGMENTS

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 #GS04T09DBC0017. 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.

³ There are approximately 40 scalar operations of various types and (5 + 18N) vector operations, where N is the number of species (e.g., 86% for H₂-CO-Air).

REFERENCES

- [1] Speth, R. L., Green, W. H., MacNamara, S., and Strang, G., "Balanced Splitting and Rebalanced Splitting," SIAM J. on Numerical Analysis 2013 51:6, 3084-3105.
- [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>.