Enabling combustion science simulations for future exascale machines

Jon S. Rood
jon.rood@nrel.gov
National Renewable Energy Laboratory
Golden, Colorado, USA

Marc T. Henry de Frahan
marc.henrydefrahan@nrel.gov
National Renewable Energy Laboratory
Golden, Colorado, USA

Marc S. Day
marcus.day@nrel.gov
National Renewable Energy Laboratory
Golden, Colorado, USA

Hariswaran Sitaraman
hariswaran.sitaraman@nrel.gov
National Renewable Energy Laboratory
Golden, Colorado, USA

Shashank Yellapantula
shashank.yellapantula@nrel.gov
National Renewable Energy Laboratory
Golden, Colorado, USA

Bruce A. Perry
bruce.perry@nrel.gov
National Renewable Energy Laboratory
Golden, Colorado, USA

Ray W. Grout
ray.grout@nrel.gov
National Renewable Energy Laboratory
Golden, Colorado, USA

Ann Almgren
asalmgren@lbl.gov
Lawrence Berkeley National Laboratory
Berkeley, California, USA

Weiqun Zhang
WeiqunZhang@lbl.gov
Lawrence Berkeley National Laboratory
Berkeley, California, USA

Jackie Chen
jhchen@sandia.gov
Sandia National Laboratories
Livermore, California, USA

CCS CONCEPTS

• **Computing methodologies → Massively parallel and high-performance simulations**.

KEYWORDS

combustion, computational fluid dynamics, adaptive mesh refinement, high performance computing, graphics processing units

ACM Reference Format:
Jon S. Rood, Marc T. Henry de Frahan, Marc S. Day, Hariswaran Sitaraman, Shashank Yellapantula, Bruce A. Perry, Ray W. Grout, Ann Almgren, Weiqun Zhang, and Jackie Chen. 2021. Enabling combustion science simulations for future exascale machines. In *Proceedings of SC '21.* ACM, New York, NY, USA, 3 pages. https://doi.org/YYY

1 SUMMARY

Combustion processes have historically been the predominant source of energy for world industry. High-fidelity simulations of turbulent combustion at exascale will play a major role in predictive design of efficient, clean engines. PeleC [3] is an application funded by the US Exascale Computing Project (ECP) for simulating combustion with complex geometries and adaptive mesh refinement (AMR) that is able to efficiently scale to the largest supercomputers currently available. PeleC is built on top of the AMReX [8, 11] framework which provides distributed parallelism, AMR, I/O, and implements an embedded boundary (EB).

PeleC is verified formally to second order accuracy for the fluid transport operators through the use of the Method of Manufactured Solutions (MMS) [9]. PeleC originally focused on CPU performance targeting the Intel Xeon Phi. Original PeleC development utilized C++ routines that called Fortran "kernels", which contain most computations. OpenMP was implemented for shared memory parallelism within MPI ranks, which favored CPUs.

PeleC allows for different chemistry mechanisms to be plugged into the code. PeleC was observed to be notably slower on Intel Xeon Phi processors (Theta [5]) than Intel Skylake processors (Eagle [2]) for all chemistry mechanisms.

With the Xeon Phi discontinued and the first exascale machines employing GPUs, PeleC required GPU capability. OpenACC was used to prototype PeleC for GPUs using the Fortran code. PeleC was also prototyped in AMReX's lambda-based C++ GPU performance portability framework.

Performance between both prototypes was similar. The AMReX C++ framework was chosen because of its advantages in portability between CPUs and multiple GPU vendors, code readability, code reduction, a single language solution, and ultimately ease of use.

SC '21, November 14–19, 2021, St Louis, MO

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/YYY

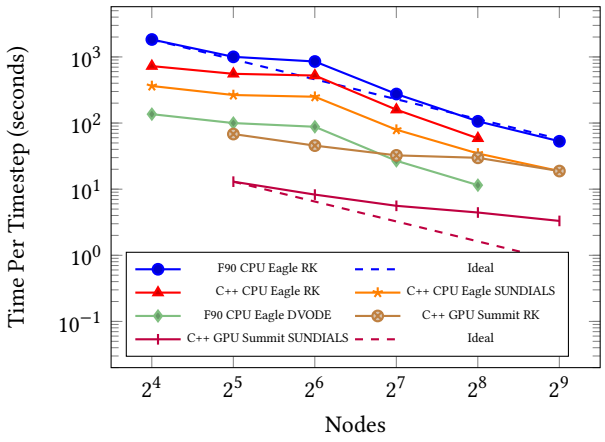

Figure 1: Strong scaling of PMF case with DRM19 chemistry on the Summit and Eagle machines. 164M cells with 2 levels of AMR. Using the Intel 2018.4 compiler on Eagle.

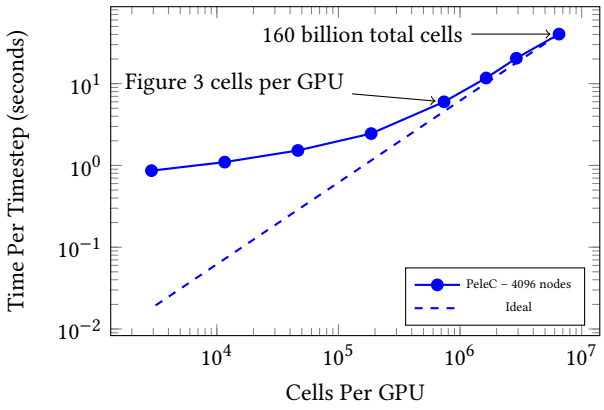

Figure 2: Strong scaling of PMF case with DRM19 chemistry on 4096 Summit nodes. Varying number of cells with 2 levels of AMR.

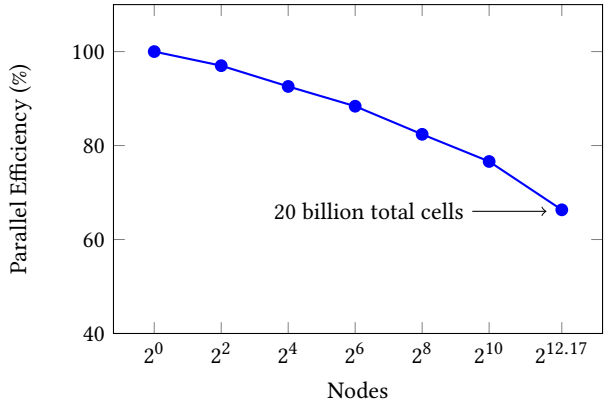

Figure 3: Weak scaling of PMF case with DRM19 chemistry. Approximately 750k cells per GPU with 2 levels of AMR. The baseline is the average time per timestep for the single node case.

Two test cases have been designed to exercise the new programming model for benchmarking. A pre-mixed flame (PMF) case and a piston bowl case.

PeleC was run on the Eagle [2] machine at National Renewable Energy Laboratory (NREL) with Intel Skylake CPUs, and the Summit [4] machine at Oak Ridge National Laboratory (ORNL) with Nvidia V100 GPUs. EB routines were observed to be insignificant in the piston bowl case and so only results from the PMF case are listed. Figure 1 shows the performance results of the GPU framework showcasing chemistry integrator options.

- Note the C++ kernels gain a 2× speedup on the CPU
- The original Fortran (F90) code on CPU with the DVODE [6] integrator is similar to the initial GPU port using PeleC's builtin explicit RK64 (RK) integrator
- Implementation of the SUNDIALS [7] integrator provides a 6× speedup over the original F90 CPU code

In Figure 2, PeleC demonstrates its ability to run a 160B cell simulation and continue to provide speedup from 6M cells per GPU down to 3k cells per GPU. Weak scaling is shown in Figure 3 where PeleC scales to a 20B cell problem using the entire Summit machine while experiencing a parallel efficiency drop of only 34%.

ACKNOWLEDGMENTS

This work was authored in part by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided by U.S. Department of Energy Office of Science and National Nuclear Security Administration. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes. This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. A portion of the research was performed using computational resources sponsored by the Department of Energy's Office of Energy Efficiency and Renewable Energy and located at the National Renewable Energy Laboratory. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

REFERENCES

[2] [n.d.]. Eagle, National Renewable Energy Laboratory. https://www.nrel.gov/hpc/eagle-system.html
[3] [n.d.]. PeleC. https://github.com/AMReX-Combustion/PeleC.
[4] [n.d.]. Summit, Oak Ridge National Laboratory. https://www.olcf.ornl.gov/summit
[5] [n.d.]. Theta, Argonne National Laboratory. https://www.alcf.anl.gov/alcf-resources/theta
[6] Alan C Hindmarsh. 1983. ODEPACK, a systematized collection of ODE solvers. *Scientific computing* (1983), 55–64.
[7] Alan C Hindmarsh, Peter N Brown, Keith E Grant, Steven L Lee, Radu Serban, Dan E Shumaker, and Carol S Woodward. 2005. SUNDIALS: Suite of nonlinear

and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)* 31, 3 (2005), 363–396.

[8] Max P. Katz, Ann Almgren, Maria Barrios Sazo, Kiran Eiden, Kevin Gott, Alice Harpole, Jean M. Sexton, Don E. Willcox, Weiqun Zhang, and Michael Zingale. 2020. Preparing Nuclear Astrophysics for Exascale. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Atlanta, Georgia) *(SC '20)*. IEEE Press, Article 91, 12 pages.

[9] Patrick J Roache. 2002. Code verification by the method of manufactured solutions. *J. Fluids Eng.* 124, 1 (2002), 4–10.

[11] Weiqun Zhang, Ann Almgren, Vince Beckner, John Bell, Johannes Blaschke, Cy Chan, Marcus Day, Brian Friesen, Kevin Gott, Daniel Graves, Max Katz, Andrew Myers, Tan Nguyen, Andrew Nonaka, Michele Rosso, Samuel Williams, and Michael Zingale. 2019. AMReX: a framework for block-structured adaptive mesh refinement. *Journal of Open Source Software* 4, 37 (May 2019), 1370. https://doi.org/10.21105/joss.01370