Enabling Combustion Science Simulations for Future Exascale Machines

Jon S. Rood[1], Marc T. Henry de Frahan[1], Marc S. Day[1], Hariswaran Sitaraman[1], Shashank Yellapantula[1], Bruce A. Perry[1], Ray W. Grout[1], Ann Almgren[2], Weiqun Zhang[2], and Jackie Chen[3]

[1]*National Renewable Energy Laboratory,* [2]*Lawrence Berkeley National Laboratory,* [3] *Sandia National Laboratories*

PeleC Application

- 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 [1] is an application funded by the US Exascale Computing Project for simulating combustion with complex geometries and adaptive mesh refinement (AMR) that is able to efficiently scale to the largest supercomputers currently available

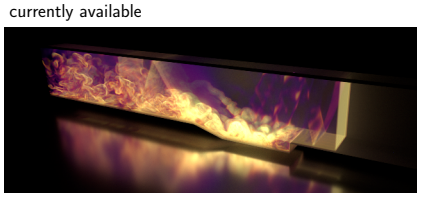

Figure 1: *Simulation of direct fuel injection in a supersonic cavity flame-holder using PeleC [2].*

- PeleC is built on top of the AMReX [3, 4] framework which provides distributed parallelism, AMR, I/O, and implements an embedded boundary (EB) as depicted in Figure 2

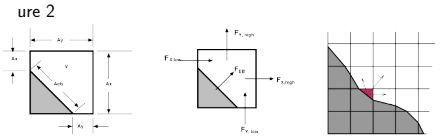

Figure 2: *EB representation of cut cell geometry in 2D. The gray region is excluded from the calculation. Figures reproduced from [5]. (a) Face areas, A. and volume, V, (b) regular fluxes, $F_{\cdot,\cdot}$, and EB fluxes, F_{EB}, (c) mass redistribution from cut cell (red) to neighbor cells.*

- PeleC is verified formally to second order accuracy for the fluid transport operators through the use of the Method of Manufactured Solutions (MMS) [6].

Original Code

- PeleC originally focused on CPU performance targeting the Intel Xeon Phi
- Original PeleC development utilized C++ routines that called Fortran "kernels", which are dense computations
- OpenMP is used for shared memory parallelism within MPI ranks, which favored CPUs
- An example of a kernel is shown in Figure 1

```
#pragma omp parallel
for (amrex::MFIter mfi(F,true); mfi.isValid(); ++mfi) {
  const amrex::Box& box = mfi.tilebox();
  amrex::Array4<amrex::Real const> const& u = U.const_array(mfi);
  amrex::Array4<amrex::Real> const& f = F.array(mfi);
  kernel(box, u, f); // Fortran kernel that can include C calls
}
```

Figure 3: *Example PeleC kernel for OpenMP C++/Fortran model.*

- PeleC allows for different chemistry mechanisms to be plugged into the code
- As seen in Figure 4 when testing mechanism runtimes, PeleC was observed to be notably slower on Intel Xeon Phi processors than Intel Skylake processors

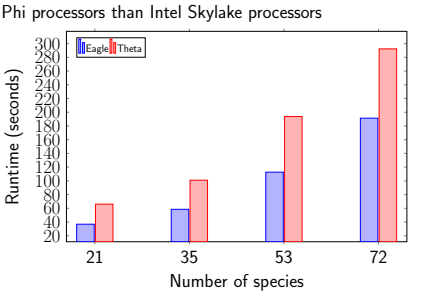

Figure 4: *Comparison of runtime for different chemistry mechanisms on Intel Skylake and Intel Xeon Phi processors for the piston bowl case (described in the Test Cases section) with 0.6M cells and one level of AMR.*

GPU Programming Models

- 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 with an example kernel in Figure 5

```
for (amrex::MFIter mfi(mf, TilingIfNotGPU()); mfi.isValid(); ++mfi) {
  const amrex::Box& bx = mfi.tilebox();
  amrex::FArrayBox& fab = mf[mfi];
  plusone_acc(BL_TO_FORTRAN_BOX(tbx), BL_TO_FORTRAN_ANYD(fab));
}

subroutine plusone_acc() ! arguments omitted
!$acc parallel loop gang vector collapse(3) default(present)
do k = lo3, hi3
  do j = lo2, hi2
    do i = lo1, hi1
      data(i,j,k) = data(i,j,k) + 1.0d
      call deep_nest_of_functions()
    end do
  end do
end do
!$acc end parallel loop
end subroutine plusone_acc
```

Figure 5: *Example PeleC kernel for OpenACC model.*

- PeleC was also prototyped in AMReX's lambda-based C++ GPU performance portability framework with an example kernel in Figure 6

```
#pragma omp parallel if (amrex::Gpu::notInLaunchRegion())
for (amrex::MFIter mfi(mf,TilingIfNotGPU()); mfi.isValid(); ++mfi) {
    const amrex::Box& bx = mfi.tilebox();
    amrex::Array4<amrex::Real> const& fab = mf.array(mfi);
    amrex::ParallelFor(bx, ncomp,
    [=] AMREX_GPU_DEVICE (int i, int j, int k, int n) {
        fab(i,j,k,n) += 1.0;
    });
}
```

Figure 6: *Example PeleC kernel using AMReX's C++ framework.*

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

Test Cases

- Two test cases have been designed to exercise the new programming model for benchmarking
- A pre-mixed flame (PMF) case illustrated in Figure 7

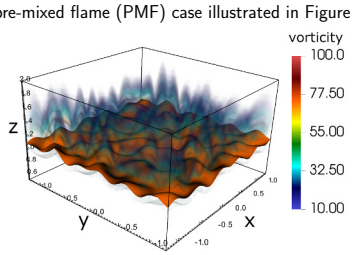

Figure 7: *Volume rendering of pre-mixed flame (PMF) case which is a thin flame sheet with corrugations. This case uses the DRM19 chemistry mechanism involving 21 species.*

- A piston bowl case shown in Figure 8 exercises EB as well as the reacting flow mechanisms

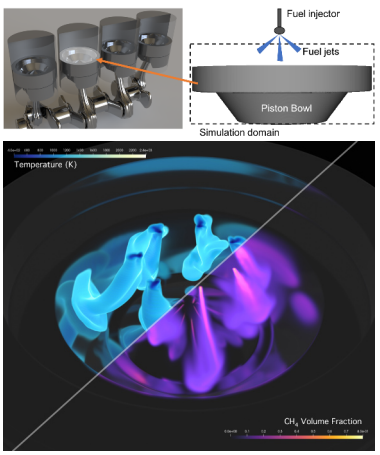

Figure 8: *Schematic of the piston bowl simulation configuration and snapshot of actual PeleC simulation.*

Performance Results

- PeleC was run on the Eagle [7] machine at National Renewable Energy Laboratory (NREL) with Intel Skylake CPUs, and the Summit [8] 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 9 shows the performance results of the GPU framework showcasing chemistry integrator options
  - Note the C++ kernels gain 2× speedup on the CPU
  - The original F90 code on CPU with the DVODE [9] integrator is similar to the initial GPU port using the explicit RK64 integrator with a 14–16× speedup
  - SUNDIALS on the GPU provides a 6× speedup over the original F90 CPU code

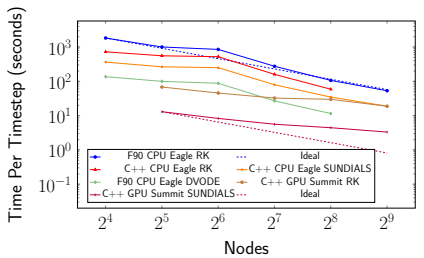

Figure 9: *Strong scaling of PMF case with DRM19 chemistry on the Summit and Eagle machines. 164M cells with 2 levels of AMR. Using the Intel compiler on Eagle.*

- In Figure 10, 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

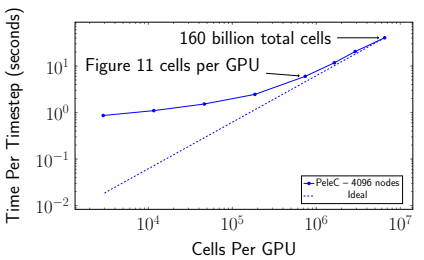

Figure 10: *Strong scaling of PMF case with DRM19 chemistry on 4096 Summit nodes. Varying number of cells with 2 levels of AMR.*

- Weak scaling is shown in Figure 11 where PeleC scales to a 20B cell problem using the entire Summit machine while experiencing a parallel efficiency drop of only 34%

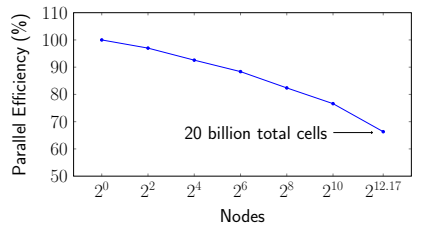

Figure 11: *Weak scaling of PMF case with DRM19 chemistry. Approximately 750k cells per GPU with 2 levels of AMR. The baseline is the average time per timestep for the single node case.*

References

[1] "PeleC." https://github.com/AMReX-Combustion/PeleC.

[2] H. Sitaraman, S. Yellapantula, M. T. Henry de Frahan, B. Perry, J. Rood, R. Grout, and M. Day, "Adaptive mesh based combustion simulations of direct fuel injection effects in a supersonic cavity flame-holder," *Combustion and Flame*, vol. 232, p. 111531, 2021.

[3] W. Zhang, A. Almgren, V. Beckner, J. Bell, J. Blaschke, C. Chan, M. Day, B. Friesen, K. Gott, D. Graves, M. Katz, A. Myers, T. Nguyen, A. Nonaka, M. Rosso, S. Williams, and M. Zingale, "AMReX: a framework for block-structured adaptive mesh refinement," *Journal of Open Source Software*, vol. 4, p. 1370, May 2019.

[4] M. P. Katz, A. Almgren, M. B. Sazo, K. Eiden, K. Gott, A. Harpole, J. M. Sexton, D. E. Willcox, W. Zhang, and M. Zingale, "Preparing nuclear astrophysics for exascale," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20, IEEE Press, 2020.

[5] "Amrex." https://amrex-codes.github.io/amrex/docs_html/index.html.

[6] P. J. Roache, "Code verification by the method of manufactured solutions," *J. Fluids Eng.*, vol. 124, no. 1, pp. 4–10, 2002.

[7] "Eagle, national renewable energy laboratory."

[8] "Summit, oak ridge national laboratory."

[9] A. C. Hindmarsh, "Odepack, a systematized collection of ode solvers," *Scientific computing*, pp. 55–64, 1983.

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering, and early testbed platforms, in support of the nation's exascale computing imperative.