

Parallel Development of Parallel Computing

Alexander Lindsay



U.S. DEPARTMENT
of **ENERGY**

Office of
Nuclear Energy

Demonstrations - Kokkos

```
lindad@cone:/data/lindad/projects/moose/test/tests/kokkos/kernels/2d_diffusion(valgrind-failures)$ MOOSE_PROFILE_BA
E_BASE=no-scatter-gpu-serial ../../../../moose_test-oprof -i kokkos_2d_diffusion_bodyforce_test.i -vec_type kok
kos -nl0_mat_type hypre -pc_type hypre -pc_hypre_boomeramg_relax_type down llscaled-Jacobi -pc_hypre_boomeramg_
relax_type up llscaled-Jacobi -pc_hypre_boomeramg_no_CF -pc_hypre_boomeramg_coarsen_type PMIS -pc_hypre_boomeram
mg_interp_type extsi --color off Mesh/square/nx=2000 Mesh/square/ny=2000 Executioner/end_time=10 Executioner/so
lve_type=P3JFNK Outputs/exodus=false Outputs/perf_graph=true Outputs/console/type=Console Outputs/console/outlie
r_variable_norms=false --n-threads=1 2>&1 | tee no-scatter-gpu-serial.log

lindad@cone:/data/lindad/projects/cpu-moose/test/tests/kokkos/kernels/2d_diffusion(cpu-moose)$ MOOSE_PROFILE_BA
SE=legacy /data/lindad/projects/cpu-moose/petsc/arch-moose/bin/mpiexec -np 30 ../../../../moose_test-oprof -i l
egacy-moose.i Outputs/console/type=Console Outputs/console/outlier_variable_norms=false --color off 2>&1 | tee
legacy.log
```

Kokkos MOOSE on 1
GPU (NVIDIA L4) vs.
classic MOOSE on 30
CPU

Kokkos 2x faster overall
solve

1.7x faster AMG

3x faster residual
evaluation (we've seen
greater speedups up to
9x with more complex
problems)

Demonstrations - MFEM

The screenshot shows a macOS terminal window with the title bar "Code" and menu items "File", "Edit", "Selection", "View", "Go", "Run", "Terminal", "Window", "Help". The terminal is connected to a remote host "moose [SSH: moose-dev-ubuntu-cone]". The left pane shows the command prompt "lindad@cone:/data/lindad/projects/moose/framework/contrib/mfem/build-oprof/examples((HEAD detached at v 4.9))\$ LD_PRELOAD=/data/lindad/gperftools/installed/lib/libprofiler.so CUPROFILE=mfem-no-print.prof ./ex26p -d cuda -no-vis -or 3 -gr 3". The right pane shows the command prompt "lindad@cone:/data/lindad/projects/moose/test/tests/kokkos/kernels/2d_diffusion(valgrind-failures)\$ MOOSE E_PROFILE_BASE=no-gen-simple-diffusion ../../../../moose_test-oprof -i kokkos_2d_diffusion_test.i Mesh/active='file' -nl0_mat_type hypr -pc_type hypr Executioner/L_tol=1e-12 -vec_type kokkos Outputs/exodus=false --disable-perflog". The bottom of the terminal shows a dock with various application icons and a status bar with system information.

MFEM: 20M dofs

MOOSE: 4M dofs



U.S. DEPARTMENT
of **ENERGY**

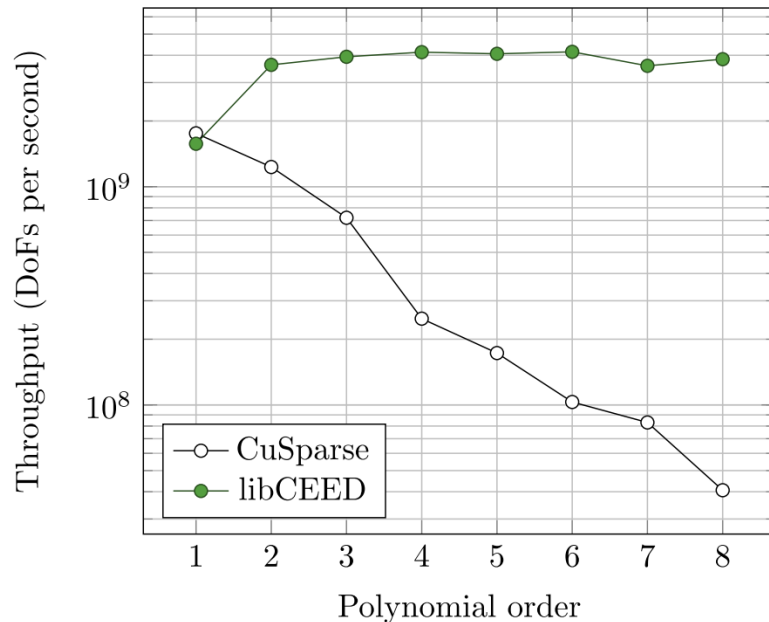
Office of
Nuclear Energy

Demonstrations - MFEM

- MOOSE/Kokkos performance during assemble/solve are strong
- libMesh/MOOSE setup costs are expensive
 - Setup costs are amortized over transients so don't be too worried!
- Increasing the problem size from 4M to 9M dofs leads to the GPU running out of memory during RAP (restriction-operator-prolongation) construction of coarse grid operators during AMG
 - Much more worrying
- How do we get 20M dofs in MFEM on the same GPU?
 - Dropping the CSR matrix and associated preconditioners for these large problems!



Motivation for high order FE and matrix-free methods



dofs per conjugate gradient solve time for a mass matrix using CSR (CuSparse) vs. matrix-free (libCEED library) on a V100 GPU. [Abdelfatth et. al.]

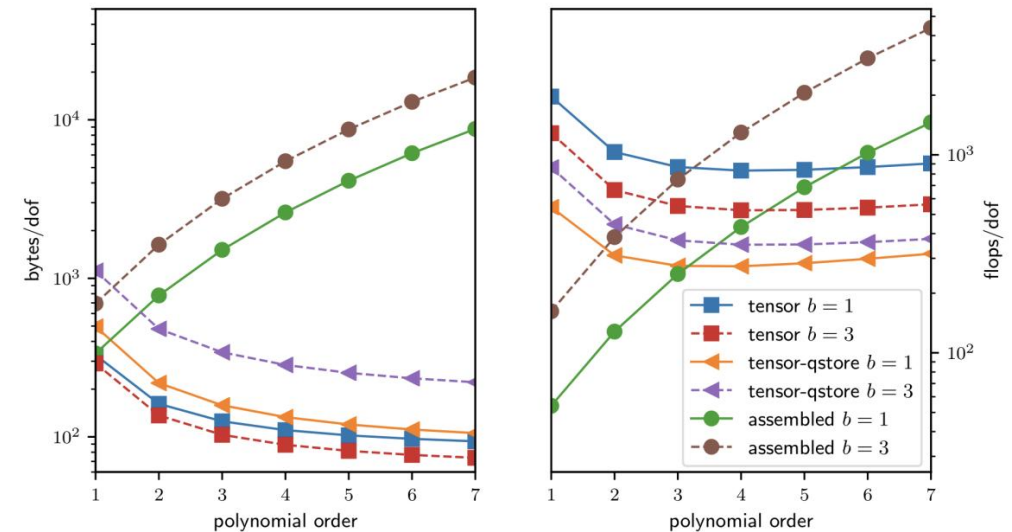


Fig. 1.1: Comparison of memory transfer and floating point operations per degree of freedom for different representations of a linear operator for a PDE in 3D with b components and variable coefficients arising due to Newton linearization of a material nonlinearity. The representation labeled as *tensor* computes metric terms on the fly and stores a compact representation of the linearization at quadrature points. The representation labeled as *tensor-qstore* pulls the metric terms into the stored representation. The *assembled* representation uses a (block) CSR format.

libCEED users manual

Motivation for high order FE and matrix-free methods

- Higher polynomial order
 - Exponential rate of convergence in smooth solution regions
 - Even when asymptotic benefit is lost near discontinuities, accuracy per degree of freedom better
- Matrix-free:
 - Higher throughput on GPU in terms of dofs/second (until saturation is reached)
 - Lower memory transfer (bytes/dof)
- Matrix-full:
 - Lower throughput on GPU
 - Higher memory transfer



So where does that leave us? How are we proceeding?

- Two great finite element libraries at our disposal
 - libMesh: years of efficient development focused on multiphysics with minimal financial support
 - MFEM: significant funding from ECP with development focused on presented concepts
- As my UK colleague said to me in discussion: “Why do you have to choose”
- So we’re not going to choose between libraries; we’re going to choose the right concepts: we need matrix-free algorithms for optimal GPU performance
- Mostly parallel acceleration frameworks--*for the moment*--libMesh/Kokkos and MFEM



Kokkos framework

- Who's it for?
 - Griffin – already has first class matrix-free sweeper algorithm
 - Finite volume based applications – Pronghorn
 - External community
- What we've done
 - Kernels, boundary conditions, materials, user objects, postprocessors, functions, auxiliary kernels
 - MetaPhysicL (MOOSE's AD library) support for Kokkos
 - Tested CUDA/Nvidia support
- What we're working on for rest of FY
 - Vector postprocessors, reporters, AD kernels
 - Test HIP/ROCm/AMD support



MFEM framework

- Who's it for?
 - Solid mechanics based applications – BISON, Grizzly, Marmot – Needs matrix-free algorithms
 - External community
- What we've done (almost all UK!)
 - Wrapping of MFEM so that users can run MOOSE input files just like they do for libMesh based sims
 - Kernels, boundary conditions, functors (MFEM coefficients), user objects, postprocessors, auxiliary kernels, MFEM-only transfers
 - Low-order refined (LOR) matrix free algorithms
- What we're working on for rest of FY
 - High order finite element and nearly matrix-free demonstration of MFEM-NEML2 integration (<https://github.com/mfem/mfem/pull/5142>)
 - Translation to MOOSE framework/solid mechanics module



How it ties together

- MultiApps and Transfers
- What we're working on for rest of FY
 - Ability to transfer data between libMesh based simulations (accelerated with Kokkos framework) and MFEM based simulations
 - Go/no-go point for MFEM within NEAMS
 - <https://github.com/idaholab/moose/issues/31982> - Currently proposed solution is construction of MFEM mesh from parent libMesh mesh to enable consistent partitioning and copy transfers
 - Will need more general solutions (hopefully our general field transfers!) for non-identical meshes which is the use case in NEAMS (different neutronics vs. fuel performance meshes etc.)





U.S. DEPARTMENT
of **ENERGY**

Office of
Nuclear Energy