On Performance Portability for Unstructured High-order Finite Element Computations

**Jed Brown** and Jeremy Thompson (CU Boulder)
Collaborators:
Jean-Sylvain Camier, Veselin Dobrev, and Tzanie Kolev (LLNL)
Misun Min (ANL)
David Medina (Two Sigma/LLNL)
Kasia Swirydowicz and Tim Warburton (Virginia Tech)
Thilina Rathnayake and Paul Fischer (University of Illinois)

SIAM Annual Meeting, 2018-07-09
Exascale Computing Project

CEED is focused on the development of next-generation discretization software and algorithms to enable efficient simulations for a wide range of science applications on future HPC systems.

- Funding: $3.0M/year, 2 labs (LLNL, ANL), 5 universities

Goals & Team

- 30+ researchers

Project Overview
Co-design Motifs

- PDE-based simulations on **unstructured grids**
- **high-order** and **spectral** finite elements
  - ✓ any order space on any order mesh ✓ curved meshes,
  - ✓ unstructured AMR ✓ optimized low-order support

10th order basis function

non-conforming AMR, 2nd order mesh

Project Overview

6th order DNS turbulence (Nek)

compressible FEM / incompressible SEM / SC

2nd order compressible shock hydro (MFEM)
High-Order Software Ecosystem

High-order Meshes
Unstructured AMR
Tensor contractions
Performance portability

PETSc
Scalable matrix-free solvers
High-Order Operator Format
General Interpolation
High-Order Visualization

PETSc

A = P^T G^T B^T D B G P

More info at: http://ceed.exascaleproject.org/fe
CEED Software Products

CEED’s library model enables ECP apps to easily take advantage of the new discretization technologies

• state-of-the-art CEED discretization libraries
  ✓ better exploit the hardware to deliver significant performance gain over conventional methods
  ✓ based on MFEM/Nek, low & high-level APIs

CEED’s proxies and general purpose libraries target ECP vendors, STs, broader community

• Ceedlings - CEED kernels, bake-off probs & miniapps
  ✓ main tools to engage vendors & external projects

• CEED broadly applicable libraries

Main deliverable: all CEED software freely available on GitHub at https://github.com/CEED
New releases: mfem-3.3, gslib, Laghos and NekCEM cceedling, …
Applicable to variety of physics

Linear, quadratic and cubic finite element spaces on curved meshes

High-order MHD
High-order radial diff.

Compressible flow (ALE, 8th order)

$H(\text{grad})$ \xrightarrow{\nabla} H(\text{curl}) \xrightarrow{\nabla \times} H(\text{div}) \xrightarrow{\nabla \cdot} L_2$

“nodes” “edges” “faces” “zones”

High-order kinematics High-order MHD High-order thermodynamics

de Rham complex

Linear, quadratic and cubic finite element spaces on curved meshes
Performance of assembled versus unassembled

- Arithmetic intensity for $Q_p$ elements
  - $< \frac{1}{4}$ (assembled), $\approx 10$ (unassembled), $\approx 5$ to 10 (hardware)
- store Jacobian information at Gauss quadrature points, can use AD
Performance versatility: $n_{1/2}$ and $t_{1/2}$

- Suppose a linear scaling algorithm
- Let $r(n)$ be the performance rate (e.g., DOF/second or GF/s) for local problem size $n = N/P$
- Let $r_{\text{max}} = \max_n r(n)$ be the peak attainable performance
- $n_{1/2} = \min\{n : r(n) \geq \frac{1}{2}r_{\text{max}}\}$
  - Local problem sizes $n < n_{1/2}$ will not yield acceptable efficiency
- $t_{1/2} = 2n_{1/2}/r_{\text{max}}$
  - Time to solution less than $t_{1/2}$ is not feasible with acceptable efficiency
2017 HPGMG performance spectra

hpgmg-fv-201706.csv

- Cori
- Sunway Taihulight
- Mira
- Hazel Hen
- Titan
- Shaheen II
- Edison

GDOF/s vs. Time (log scale)
CEED Bake-Off Problems

**BP1**: Solve \( \{Bu=f\} \), where \( \{B\} \) is the mass matrix.

**BP2**: Solve the vector system \( \{Bu_i=f_i\} \) with \( \{B\} \) from BP1.

**BP3**: Solve \( \{Au=f\} \), where \( \{A\} \) is the Poisson operator.

**BP4**: Solve the vector system \( \{Au_i=f_i\} \) with \( \{A\} \) from BP3.

- **Range of polynomial orders**: \{p=1, 2,...,8\}, at least.
- **Cover range of sizes**: from 1 element/MPI rank up to the memory limit.
- **BP1 and BP2** are relevant for many hyperbolic substeps in transport problems. BP3 and BP4 reflect pressure, momentum, and diffusion updates in fluid/thermal transport.
- **Vector forms BP2 and BP4** reveal benefits of increased *data reuse* and of *amortized communication overhead*.

**Benchmark repo**: [https://github.com/CEED/benchmarks](https://github.com/CEED/benchmarks)
Figure: BP1 results of Nek5000 (left), MFEM (center), and deal.ii (right) on BG/Q with varying polynomial order \((p = 1, \ldots, 16)\) with the number of quadrature points \((q = p + 2)\). The number cpu cores \(P = 8, 192\).
BP1 on KNL: Nek5000 and MFEM

Nek5000 $n_{1/2} = 15k, t_{1/2} = 150\mu s$

- BG/Q has similar performance

MFEM $n_{1/2} = 10k, t_{1/2} = 400\mu s$
Lightweight Performance Portability

CEED/OCCA is an open-source library that provides an unified API for programming different types of devices, including CPUs, GPUs, Intel’s Xeon Phi, FPGAs.

Features:

• Supported on many languages, such as C++, C, and Fortran
• *JIT compilation for kernels*
• Single kernel language for all backends (OKL)
• Currently supports Serial, OpenMP, CUDA, and OpenCL backends. Works with MPI
• MIT License, [http://www.libocca.org](http://www.libocca.org)
• Extensible backend API, allowing for future features. For example, support for unified memory in CUDA and mapped memory in OpenCL
OCCA performance on Summit (V100)

Figure: BK1 and BK3 V100 performance: TFLOPS versus problem size $n$ for different polynomial orders, $N$. Operating on E-vectors (does not include element restriction $E, E^T$)
Batched Computing Technology

- Matrix-free basis evaluation needs efficient tensor contractions, e.g.,
  \[ C_{i_1,i_2,i_3} = \sum_k A_{k,i_1} B_{k,i_2,i_3} \]

- **CEED/MAGMA** designed batched methods to split the computation in many small high-intensity GEMMs, grouped together (batched) for efficient execution:
  \[ \text{Batched}\{ \ C_{i_3} = A^T B_{i_3}, \text{ for range of } i_3 \} \]

- Developed techniques needed for autotuning, code inlining, code generation (reshapes, etc.), algorithmic variants for different architectures.
- Achieve 90+% of theoretically derived peaks.
- Significantly outperform vendor libraries.
- Released through MAGMA.
MPICH CH4: lightweight device layer

- CH4: faster offload, better fast path/inlining/IPO

Nek5000 Mass-Matrix Inversion, Lite & Std MPI.

Nek5000 Mass-Matrix Inversion (Lite/Std)
libCEED: Code for Efficient Extensible Discretization

- BSD-2 license, C library with Fortran interface
- Releases: v0.1 (January), v0.2 (March), v0.3 (imminent)
- Purely algebraic interface
- Extensible backends
  - CPU: reference, vectorized
  - OCCA (just-in-time compilation): CPU, OpenMP, OpenCL, CUDA
  - MAGMA
- Platform for collaboration with vendors
- Minimal assumptions about execution environment, parallel decomposition
- Primary target: high order finite element methods
  - $H^1$, $H(\text{div})$, $H(\text{curl})$
  - also of interest to spectral difference, etc.
  - Exploit tensor product structure when possible
\[ A = \mathbf{P}^T \mathbf{E}^T \mathbf{B}^T \mathbf{D} \mathbf{B} \mathbf{E} \mathbf{P} \]
Quadrature Function

\[ v^T F(u) \sim \int_\Omega \nu \cdot f_0(u, \nabla u) + \nabla \nu \cdot f_1(u, \nabla u) \quad v^T Jw \sim \int_\Omega \begin{bmatrix} v \\ \nabla v \end{bmatrix}^T \begin{bmatrix} f_{0,0} & f_{0,1} \\ f_{1,0} & f_{1,1} \end{bmatrix} \begin{bmatrix} w \\ \nabla w \end{bmatrix} \]

\[ u_e = B_{\mathcal{E}_e} u \quad \nabla u_e = \frac{\partial X}{\partial x} B_{\nabla} \mathcal{E}_e u \]

\[ Jw = \sum_e \mathcal{E}_e^T \begin{bmatrix} B \\ B_{\nabla} \end{bmatrix}^T \begin{bmatrix} I \\ \left( \frac{\partial X}{\partial x} \right)^T \end{bmatrix} W_q \begin{bmatrix} f_{0,0} & f_{0,1} \\ f_{1,0} & f_{1,1} \end{bmatrix} \begin{bmatrix} I \\ \left( \frac{\partial X}{\partial x} \right) \end{bmatrix} \begin{bmatrix} B \\ B_{\nabla} \end{bmatrix} \mathcal{E}_e W \]

| B and \( B_{\nabla} \) are tensor contractions – independent of element geometry |
| Choice of how to order and represent gathers \( \mathcal{E} \) and scatters \( \mathcal{E}^T \) |
| Who computes the metric terms and other coefficients? |
| Similar for Neumann/Robin and nonlinear boundary conditions |
Quadrature Functions

- Multiple inputs and outputs
- Independent operations at each of $Q$ quadrature points
- Ordering and number of elements not specified

```c
int L2residual(void *ctx, CeedInt Q,
                const CeedScalar *const in[],
                CeedScalar *const out[])
{
    const CeedScalar *u = in[0], *rho = in[1], *target = in[2];
    CeedScalar *v = out[0];
    for (CeedInt i=0; i<Q; i++)
        v[i] = rho[i] * (u[i] - target[i]);
    return 0;
}
```
Element restriction $\mathcal{E}_e$

- Conforming homogeneous mesh: boolean matrix with homogeneous block size
- Non-conforming mesh: anchored rows have linear combination
- Nek5000-style E-vector: indexed identity
- libCEED backends are allowed to reorder, compress, etc.
- May be applied all at once or in batches
libCEED Operator

\[ A = \mathcal{P}^T \mathcal{E}^T \mathcal{B} \mathcal{D} \mathcal{B} \mathcal{E} \mathcal{P} \]

- element restriction \( \mathcal{E} \), basis \( \mathcal{B} \), quadrature function \( \mathcal{D} \)

```c
CeedOperatorCreate(ceed, qf_L2residual, &op);
CeedOperatorSetField(op, "u", E, Basis, CEED_VECTOR_ACTIVE);
CeedOperatorSetField(op, "rho", CEED_RESTRICTION_IDENTITY, CEED_BASIS_COLOCATED, rho);
CeedOperatorSetField(op, "target", CEED_RESTRICTION_IDENTITY, CEED_BASIS_COLOCATED, target);
CeedOperatorSetField(op, "v", E, Basis, CEED_VECTOR_ACTIVE);
```
Vectorization techniques

- Vectorize within a single high-order element
  - Minimal working set (as small as one element)
  - Specialized implementation for different degree/# quadrature points
  - Hard to avoid cross-lane operations at modest degree
  - Nek5000

- Vectorize across elements in batches $[i, j, k, e]$
  - Working set has at least vector length number of elements (e.g., 8)
  - Generic implementation is easy to optimize; no cross-lane operations
  - HPGMG-FE, Deal.II (Kronbichler and Kormann), MFEM (new)
MFEM vectorization performance

Figure: Internal versus external element vectorization for BP1.
HPGMG: a benchmark for supercomputers

- https://hpgmg.org
- Mark Adams, Sam Williams (finite-volume), Jed (finite-element), John Shalf, Brian Van Straalen, Erich Strohmeier, Rich Vuduc
- Annual BoFs at Supercomputing since 2014
- Implementations
  - **Finite Volume** memory bandwidth intensive, simple data dependencies, 4th order
  - **Finite Element** compute- and cache-intensive, vectorizes, overlapping writes
- Full multigrid, well-defined, scale-free problem
- Matrix-free operators, Chebyshev smoothers
Full Multigrid (FMG): Prototypical Fast Algorithm

- start with coarse grid
- truncation error within one cycle
- about five work units for many problems
- no “fat” left to trim – robust to gaming
- distributed memory – restrict active process set using Z-order
  - $\mathcal{O}(\log^2 N)$ parallel complexity stresses network
- scale-free specification
  - no mathematical reward for decomposition granularity
  - don’t have to adjudicate “subdomain”
HPGMG-FE on Edison, SuperMUC, Titan

Titan >200ms
variability
1.6B
155B
309B
12.9B

Climate SYPD goal

DOF/s

TFlop/s

Solve time (s)
Outlook

- libCEED is interested in contributors and friendly users
- GPU performance optimizations in progress
- Cache versus vectorization tradeoffs
  - Backends should automatically choose internal versus external vectorization
  - Choice depends on architecture, element size, number of fields
- Throughput versus latency optimizations
- Even/odd performance optimization
- Incorporate algorithmic differentiation
- Developing exchange/storage interfaces for high-order fields
- Many other activities to improve high order ecosystem