# Matrix Free P-Multigrid with libCEED and PETSc

Jeremy L Thompson

University of Colorado Boulder

*jeremy.thompson@colorado.edu*

June 28, 2019

# libCEED Team

Developers: Jed Brown[1], Jeremy Thompson[1]
Valeria Barra[1], Tzanio Kolev[2], Jean-Sylvain Camier[2], Veselin Dobrev[2], Yohann Doudouit[2], Tim Warburton[3], David Medina[4], & Thilina Rathnayake[5]

Grant: Exascale Computing Project (17-SC-20-SC)

1: University of Colorado, Boulder
2: Lawrence Livermore National Laboratory
3: Virginia Polytechnic Institute and State University
4: OCCA
5: University of Illinois, Urbana-Champaign

# Overview

libCEED is an extensible library that provides a portable algebraic interface and optimized implementations of high-order operators

We have optimized implementations targeting CPU and GPU

We investigate a p-multigrid example with PETSc PCMG

# Overview

# Center for Efficient Exascale Discretizations

DoE exascale co-design center

- Design discretization algorithms for exascale hardware that deliver significant performance gain over low order methods
- Collaborate with hardware vendors and software projects for exascale hardware and software stack
- Provide efficient and user-friendly unstructured PDE discretization component for exascale software ecosystem

# Matrix Free

libCEED design approach:

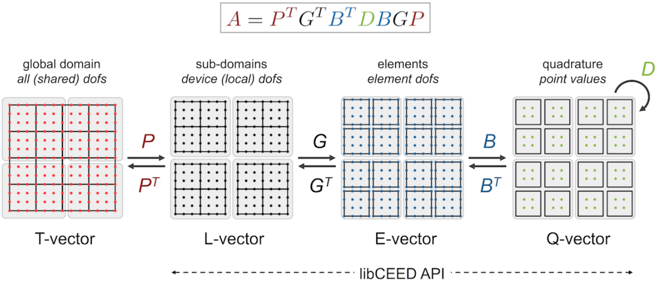
- Avoid global matrix assembly
- Map each element to reference element
- Geometry data computed on the fly or precomputed
- Easy to parallelize across hetrogeneous nodes

# libCEED

libCEED provides multiple backend implementations

- CPU
  - Pure C
  - Advanced Vector Instructions
  - LIBXSMM
- GPU
  - Pure CUDA
  - OCCA
  - MAGMA

# libCEED




$$A_L = G^T B^T D B G$$

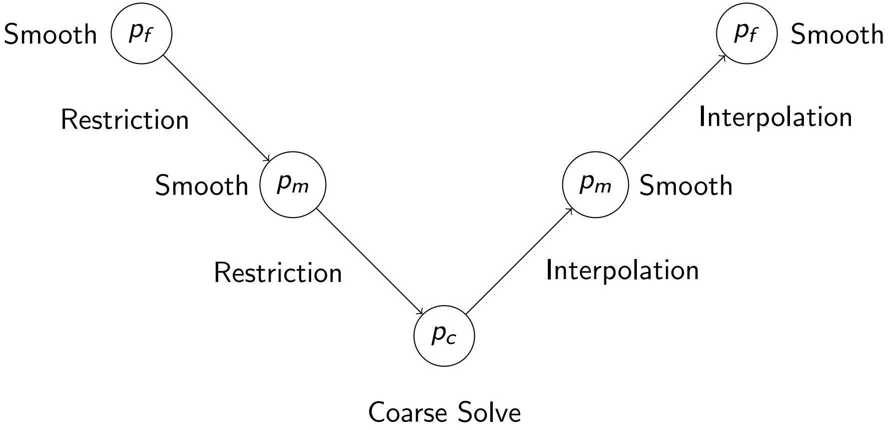
- $G$ - CeedElemRestriction, local gather/scatter
- $B$ - CeedBasis, provides basis operations such as interp and grad
- $D$ - CeedQFunction, representation of PDE at quadrature points
- $A_L$ - CeedOperator, aggregation of Ceed objects for local action of operator

# PETSc PCMG

- PCMG - PETSc geometric multigrid preconditioner
- Requires several operators from the user
  - Restriction operator
  - Interpolation operator
  - Smoother
  - Coarse grid solver

# PETSc PCMG

3 level multigrid with PCMG



# libCEED Operators - Diffusion

Solving the 2D Poisson problem: $-\Delta u = f$

Weak Form: $\int \nabla v \nabla u = \int v f$

- General libCEED Operator
  $A_L = G^T B^T D B G$

- Diffusion Operator
  $A_L = G^T \hat{D}_{2d}^T D \hat{D}_{2d} G$

  where $D$ is block diagonal by quadrature point:

$$D_i = \left( w_i \det J_{geo} \right) J_{geo}^{-T} J_{geo}^{-1} \text{ and } J_{geo} = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial s} \end{bmatrix}$$

# libCEED Operators - Diffusion

Solving the 2D Poisson problem: $-\Delta u = f$
Weak Form: $\int \nabla v \nabla u = \int vf$

- General libCEED Operator
  $A_L = G^T B^T D B G$

- Diffusion Operator
  $A_L = G^T \hat{D}_{2d}^T D \hat{D}_{2d} G$

- Computationaly Efficient Form

$$A_L = G^T \begin{bmatrix} \hat{D}^T \otimes \hat{J}^T & \hat{J}^T \otimes \hat{D}^T \end{bmatrix} D \begin{bmatrix} \hat{D} \otimes \hat{J} \\ \hat{J} \otimes \hat{D} \end{bmatrix} G$$

# libCEED Operators - Diffusion

Solving the 2D Poisson problem: $-\Delta u = f$
Weak Form: $\int \nabla v \nabla u = \int vf$

- General libCEED Operator
  $A_L = G^T B^T D B G$

- Diffusion Operator
  $A_L = G^T \hat{D}_{2d}^T D \hat{D}_{2d} G$

- Computationaly Efficient Form
  $A_L = G^T \left( \hat{J}^T \otimes \hat{J}^T \right) \begin{bmatrix} \tilde{D}^T \otimes I & I \otimes \tilde{D}^T \end{bmatrix} D \begin{bmatrix} \tilde{D} \otimes I \\ I \otimes \tilde{D} \end{bmatrix} \left( \hat{J} \otimes \hat{J} \right) G$
  where $\hat{D} = \tilde{D} \hat{J}$

# libCEED Operators - Restriction

Restriction / Interpolation is largely a basis operation

- General libCEED Operator
  $A_L = G^T B^T D B G$
- Restriction / Interpolation Operator
  $A_L = G_c^T I I B_{ftoc} G_f$
- Computationaly Efficient Form
  $A_L = G_c^T \left( \hat{J}_{ftoc} \otimes \hat{J}_{ftoc} \right) G_f$

# libCEED Operators - Smoothing

For smoothing, we use a libCEED diffusion operator with KSPCHEBYCHEV

- General libCEED Operator
  $A_L = G^T B^T D B G$
- Diffusion Operator
  $A_L = G^T \hat{D}_{2d}^T D \hat{D}_{2d} G$
- Computationaly Efficient Form
  $$A_L = G^T \left( \hat{J}^T \otimes \hat{J}^T \right) \begin{bmatrix} \tilde{D}^T \otimes I & I \otimes \tilde{D}^T \end{bmatrix} D \begin{bmatrix} \tilde{D} \otimes I \\ I \otimes \tilde{D} \end{bmatrix} \left( \hat{J} \otimes \hat{J} \right) G$$

# QFunction Definition

General libCEED QFunction:

$$v_q = D u_q$$

2D Diffusion QFunction:

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} D_{00} & D_{01} \\ D_{01} & D_{11} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Code:

```
CeedQFunctionCreateInterior(ceed, 1, Diff,
                            __FILE__multigrid.c:Diff, &qf_apply);
CeedQFunctionAddInput(qf_apply, "u", 1, CEED_EVAL_GRAD);
CeedQFunctionAddInput(qf_apply, "geo", 3, CEED_EVAL_NONE);
CeedQFunctionAddOutput(qf_apply, "v", 1, CEED_EVAL_GRAD);
```

# Operator Definition

General libCEED Operator:

$$v_L = A_L u_L$$

2D Diffusion Operator:

$$A_L = \mathbf{G}^{\mathbf{T}} B^T D B \mathbf{G}$$

Code:

```
CeedOperatorCreate(ceed, qf_apply, NULL, NULL, &op_apply);
CeedOperatorSetField(op_apply, "u", Erestrictu, CEED_TRANSPOSE,
                     basisu, CEED_VECTOR_ACTIVE);
CeedOperatorSetField(op_apply, "geo",Erestrictqdi,CEED_NOTRANSPOSE,
                     CEED_BASIS_COLLOCATED, geo);
CeedOperatorSetField(op_apply, "v", Erestrictu, CEED_TRANSPOSE,
                     basisu, CEED_VECTOR_ACTIVE);
...
CeedOperatorApply(op_apply, xloc, yloc, CEED_REQUEST_IMMEDIATE);
```

# Performance

- 3D Poisson Problem
- Test run on personal computer
- Mesh
  - $8^3$ GLL points per element
  - Quadrature on $9^3$ GL points per element
  - Cube with 30 elements, $11,880$ DoFs
- Unpreconditioned
  - 5.0057e-07 — $||\cdot||_\infty$ Error
  - 119 — CG iterations
  - 0.2148 million — CG DoFs/sec
  - 13.5436 sec — CG solve time
- P-Multigrid
  - 5.0059e-07 — $||\cdot||_\infty$ Error
  - 18 — CG iterations
  - 0.0795 million — CG DoFs/sec
  - 4.1058 sec — CG solve time

# Performance - Highlights

- Significantly decreased number of iterations
- Iterations slower than expected
- Want to decrease iteration time with lighter preconditioner
- Caveats:
  - Small mesh run on laptop for demo purposes
  - Need minor PETSc code adjustments to run well on GPU

# Future Work

- Further performance tuning (GPU and CPU)
- Unstructured mesh examples (with AMG coarse solve)
- Expanded set of non-linear examples
- Preconditioning based on libCEED operator decomposition
- Efficient diagonal computation for preconditioning
- Algorithmic differentiation of user quadrature functions
- We invite contributors and friendly users

# Questions?

Advisors : Jed Brown[1] & Daniel Appelö[1]

Collaborators: Valeria Barra[1], Oana Marin[2], Tzanio Kolev[3], Jean-Sylvain Camier[3], Veselin Dobrev[3], Yohann Doudouit[3], Tim Warburton[4], David Medina[5], & Thilina Rathnayake[6]

Grant: Exascale Computing Project (17-SC-20-SC)

1: University of Colorado, Boulder
2: Argonne National Laboratory
3: Lawrence Livermore National Laboratory
4: Virginia Polytechnic Institute and State University
5: OCCA
6: University of Illinois, Urbana-Champaign

# Matrix Free P-Multigrid with libCEED and PETSc

Jeremy L Thompson

University of Colorado Boulder

*jeremy.thompson@colorado.edu*

June 28, 2019