On Performance and Portability for Generic Finite Element Interfaces

Jeremy L Thompson

University of Colorado Boulder

jeremy.thompson@colorado.edu

March 2, 2018

Overview

A global sparse matrix is no longer a good representation of a high-order linear operator

libCEED is an extensible library that provides a portable algebraic interface and optimized implementations

We have preliminary results comparing performance to native implementations in production software

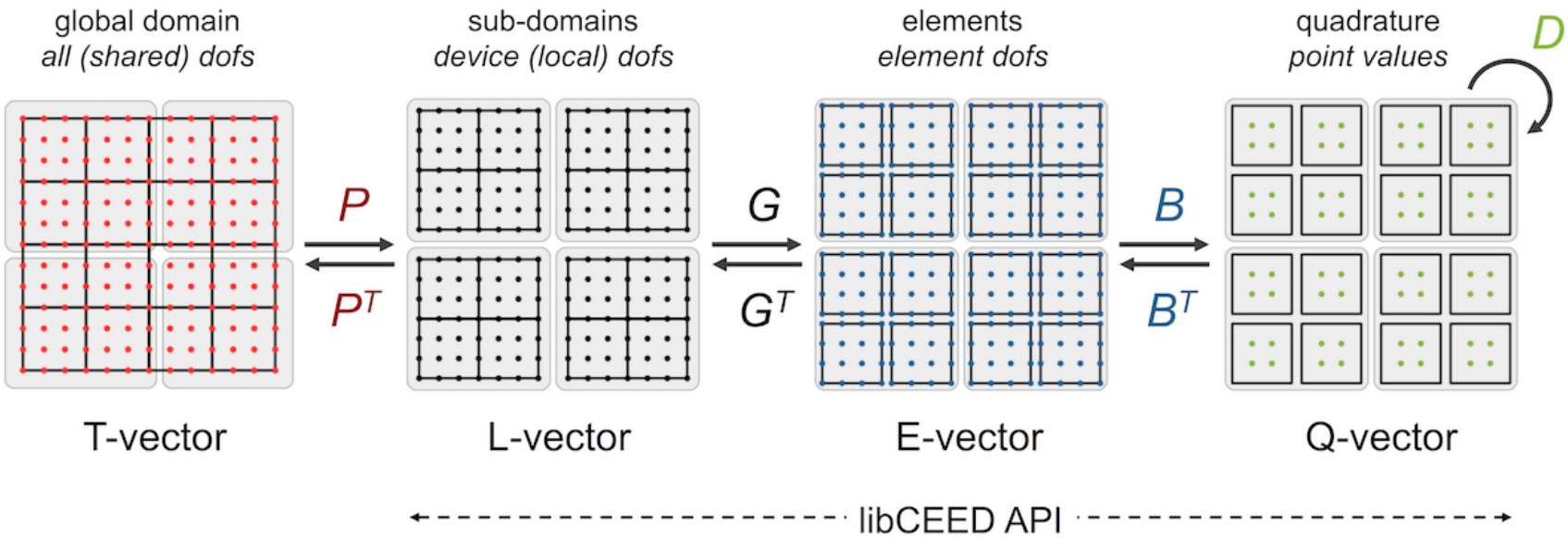
Overview

Weak Form and Finite Elements

- Strong Form of PDE: $A_s u = f$
- Weak Form of PDE: $\int_\Omega Auv = \int_\Omega fv$
- v are test functions defined on each element,
  yields system of equations
- Operator can be decomposed algebraically

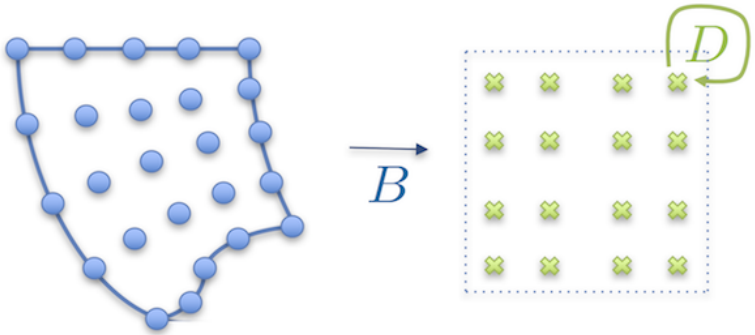
Operator Decomposition

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



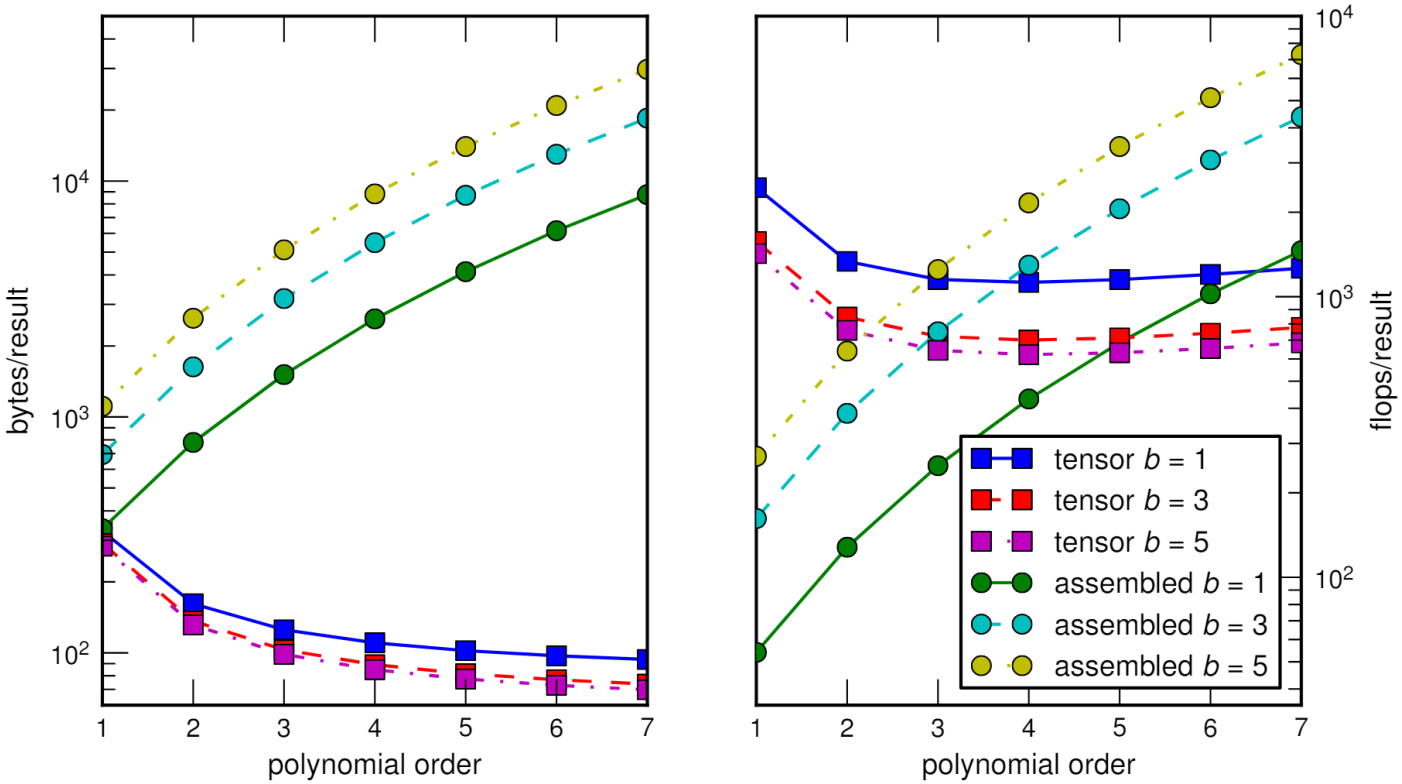
Matrix Free Implementation

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



- Avoid global matrix assembly
- Map each element to reference element
- Only store map to reference, action on reference
- Easy to parallelize across nodes

Assembled Matrix Cost!



Memory bandwidth and ops per dof to apply a Jacobian from Q_k discretization of a b-variable PDE system using an assembled matrix versus matrix-free exploiting the tensor product structure

libCEED API

- Provides on-device operator implementation
- Easy to incorporate into existing code
- Supports multiple types of computational devices
  GPUs, CPUs, etc
- Multiple extensible implementations
  Reference on CPUs, OCCA on GPUs

API Objects

- G - CeedRestriction
  Restrict to single element

- B - CeedBasis
  Actions on basis such as interpolation, gradient, divergence, curl

- D - CeedQFunction
  Operator action at quadrature points to include coefficient functions

Device Level Operator

- $L = G^T B^T D B G$ - CeedOperator
- libCEED objects are combined to create a CeedOperator
- CeedOperator gives operator action for elements on device
- User code responsible for communication between devices

  $A = P^T L P$

Benefits

- Extensible library
- Lower memory transfer, no sparse matrix
- Implementations for multiple devices and backends
- libCEED optimization can benefit all operators
  Tensor contraction, basis application, etc

Standalone Implementation

```
// Create the mass operator.
CeedOperator oper;
CeedOperatorCreate(ceed, sol_restr, sol_basis,
                   apply_qfunc, NULL, NULL, &oper);
```

...

```
// Apply the mass operator: 'u' -> 'v'.
CeedOperatorApply(oper, qdata, u, v,
                  CEED_REQUEST_IMMEDIATE);
```

MFEM

```
/// Wrapper for a mass CeedOperator as an
/// mfem::Operator
class CeedMassOperator : public mfem::Operator
 protected:
  const mfem::FiniteElementSpace *fes;
  CeedOperator build_oper, oper;
  CeedBasis basis, mesh_basis;
  CeedElemRestriction restr, mesh_restr;
  CeedQFunction apply_qfunc, build_qfunc;
  CeedVector node_coords, qdata;
```

Nek5000

```
      subroutine ceed_axhm1(pap,ap1,p1,h1,h2,ceed,op_mass,
     $ vec_ap1,vec_p1,vec_qdata)

      include 'ceedf.h'

c Vector conjugate gradient matvec for solution of
c uncoupled Helmholtz equations

      include 'SIZE'
      include 'TOTAL'
      ...
      call ceedvectorsetarray(vec_p1,ceed_mem_host,
     $ ceed_use_pointer, p1,err)
      call ceedoperatorapply(op_mass,vec_qdata,vec_p1,vec_ap1,
     $ ceed_request_immediate,err)
      call ceedvectorgetarray(vec_ap1,ceed_mem_host,ap1,err)
```

PETSc

```
user->op = op_mass;
user->qdata = qdata;

ierr = MatCreateShell(comm, mdof[0]*mdof[1]*mdof[2],
       mdof[0]*mdof[1]*mdof[2],
       PETSC_DECIDE, PETSC_DECIDE, user, &mat);
CHKERRQ(ierr);
ierr = MatShellSetOperation(mat, MATOP_MULT
       (void(*)(void))MatMult_Mass); CHKERRQ(ierr);

...

ierr = KSPSetFromOptions(ksp); CHKERRQ(ierr);
ierr = KSPSetOperators(ksp, mat, mat); CHKERRQ(ierr);
ierr = KSPSolve(ksp, rhs, X); CHKERRQ(ierr);
```

Nek5000

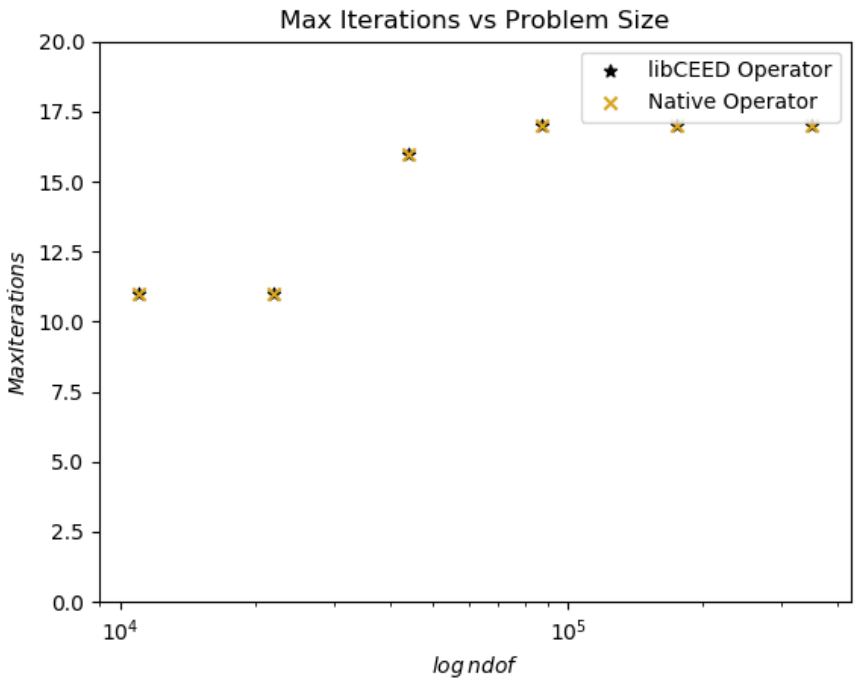
Problem: $\nabla u = f$
CEED Benchmark Problem 1

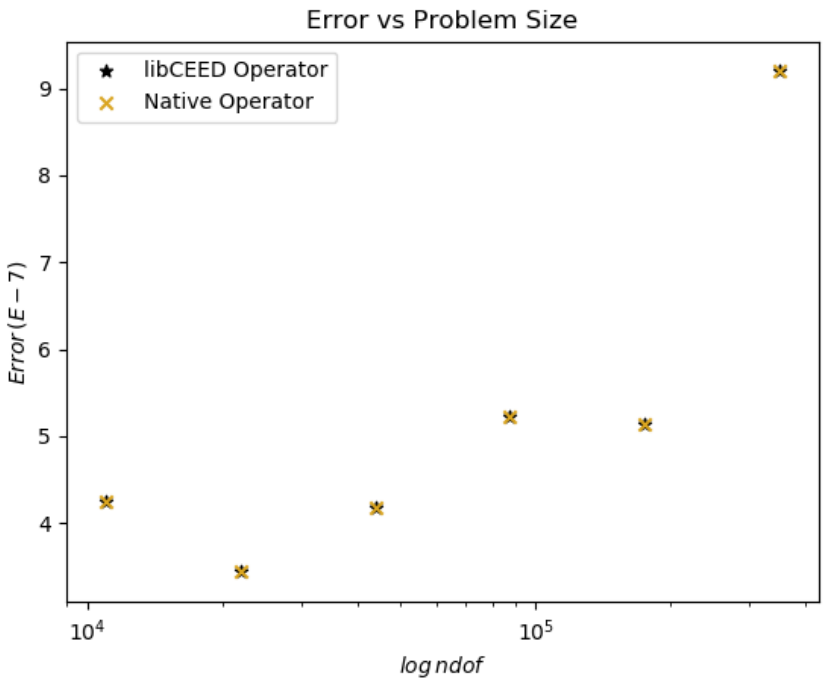
Domain: 3D Cube
Elements: Hexagonal
Number of Elements: 2^n
Shape Function Order: 7
Quadrature Points: 9

Computer: CU Boulder Summit

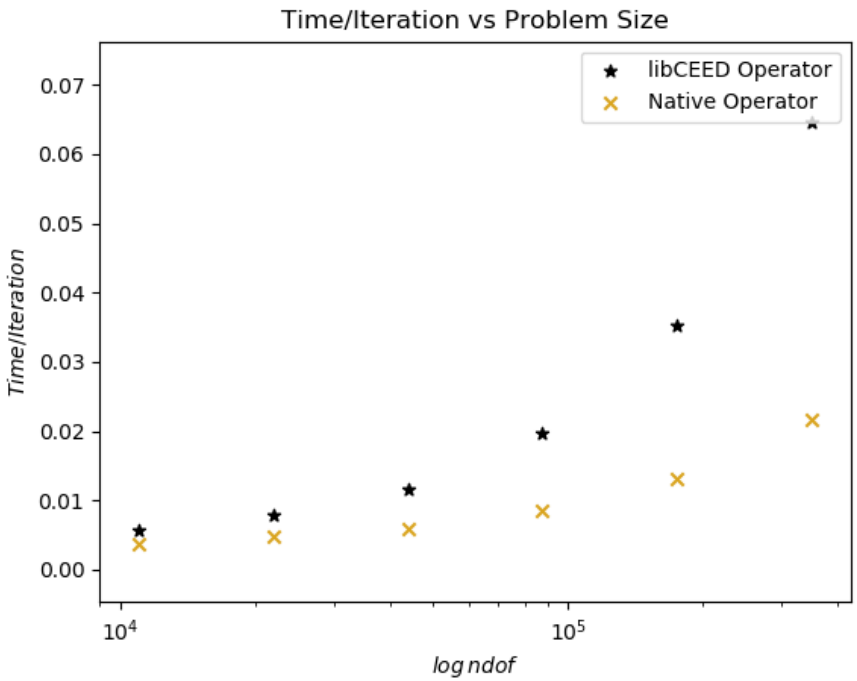
Nodes: 1
CPUs: Intel Xeon "Haswell"
Processors: 32
Compiler: Intel/17.0.0
MPI: Intel/2017.0.098

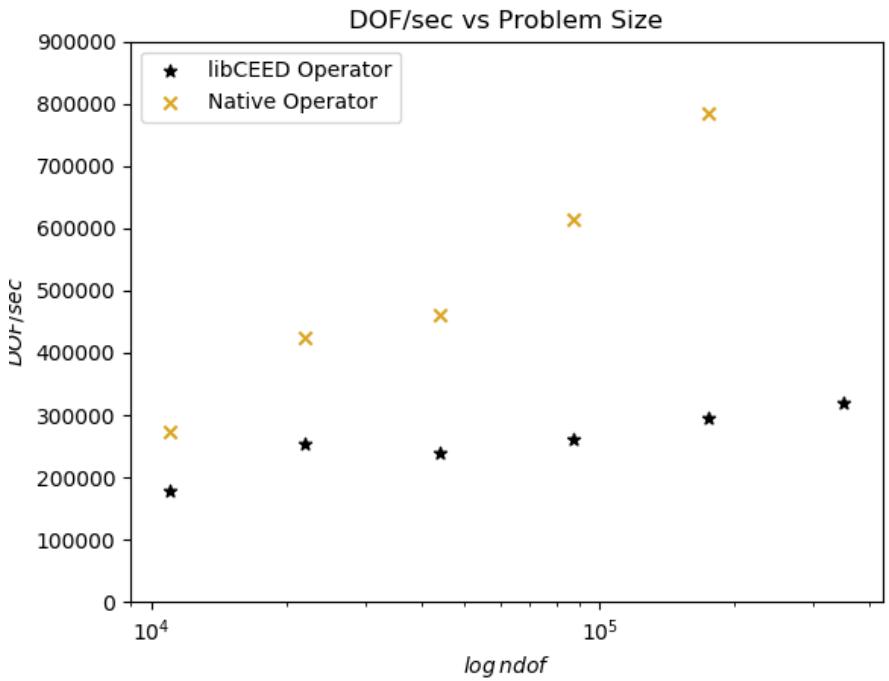
Nek5000 - The Good News





Nek5000 - The Bad News





Future Work

- Optimize reference implementation, tensor contraction
- Create library of user quadrature functions
- Create additional backends
- Compare libCEED operators to native implementation in a wider range of production software

Questions?

Advisor: Jed Brown[1]

Collaborators: Jean-Sylvain Camier[2], Tzanio Kolev[2], Veselin Dobrev[2], & Thilina Rathnayake[3]

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

1: University of Colorado, Boulder
2: Lawrence Livermore National Laboratory
3: University of Illinois, Urbana-Champaign

On Performance and Portability for Generic Finite Element Interfaces

Jeremy L Thompson

University of Colorado Boulder

jeremy.thompson@colorado.edu

March 2, 2018