Stokes Preconditioning on a GPU

Matthew Knepley\textsuperscript{1,2}, Dave A. Yuen, and Dave A. May

\textsuperscript{1}Computation Institute
University of Chicago

\textsuperscript{2}Department of Molecular Biology and Physiology
Rush University Medical Center

AGU ’09
San Francisco, CA December 15, 2009
Collaborators

- **Prof. Dave Yuen**
  - Dept. of Geology and Geophysics, University of Minnesota

- **Dr. David May**, developer of BFBT (in PETSc)
  - Dept. of Earth Sciences, ETHZ

- **Felipe Cruz**, developer of FMM-GPU
  - Dept. of Applied Mathematics, University of Bristol

- **Prof. Lorena Barba**
  - Dept. of Mechanical Engineering, Boston University
Outline

1. 5 Slide Talk
2. What are the Problems?
3. Can we do Better?
4. Advantages and Disadvantages
5. What is Next?
BFBT preconditions the Schur complement using

\[ S_b^{-1} = L_p^{-1} G^T KGL_p^{-1} \]  \hspace{1cm} (1)

where \( L_p \) is the Laplacian in the pressure space.
The current BFBT code is limited by

- **Bandwidth constraints**
  - Sparse matrix-vector product
  - Achieves at most 10% of peak performance

- **Synchronization**
  - GMRES orthogonalization
  - Coarse problem

- **Convergence**
  - Viscosity variation
  - Mesh dependence
Use a **Boundary Element Method**, for the Laplace solves in BFBT, accelerated by **FMM**.
Missing Pieces

- **BEM discretization and assembly**
  - Matrix-free operator application using the Fast Multipole Method
  - Overcomes bandwidth limit, 480 GF on an NVIDIA 1060C GPU
  - Overcomes coarse bottleneck by overlapping direct work

- **Solver for BEM system**
  - Same total work as FEM due to well-conditioned operator
  - Possibility of multilevel preconditioner (even better)

- **Interpolation between FEM and BEM**
  - Boundary interpolation just averages
  - Can again use FMM for interior
Direct Fast Method for Variable-Viscosity Stokes

- Complexity not currently precisely quantified
  - We would like a given number of flops/digit of accuracy

- Brute Force
  - Use BEM to compute layers between regions of constant viscosity
  - Better conditioned, but not direct

- Elegant method should be possible
  - The operator is pseudo-differential
  - “Kernel-independent” FMM exists
Direct Fast Method for Variable-Viscosity Stokes

- Complexity not currently precisely quantified
  - We would like a given number of flops/digit of accuracy

- Brute Force
  - Use BEM to compute layers between regions of constant viscosity
  - Better conditioned, but not direct

- Elegant method should be possible
  - The operator is pseudo-differential
  - “Kernel-independent” FMM exists
Direct Fast Method for Variable-Viscosity Stokes

- Complexity not currently precisely quantified
  - We would like a given number of flops/digit of accuracy

- Brute Force
  - Use BEM to compute layers between regions of constant viscosity
  - Better conditioned, but not direct

- Elegant method should be possible
  - The operator is pseudo-differential
  - "Kernel-independent" FMM exists
Outline

1. 5 Slide Talk

2. What are the Problems?
   - Bandwidth
   - Synchronization
   - Convergence

3. Can we do Better?

4. Advantages and Disadvantages

5. What is Next?
The current BFBT code is limited by

- Bandwidth constraints
- Synchronization
- Convergence
Outline

What are the Problems?

- Bandwidth
- Synchronization
- Convergence
Bandwidth

Small bandwidth to main memory can limit performance

- Sparse matrix-vector product
- Operator application
- AMG restriction and interpolation
STREAM Benchmark

Simple benchmark program measuring **sustainable** memory bandwidth

- Protoypical operation is Triad \((WAXPY)\): \(w = y + \alpha x\)
- Measures the memory bandwidth bottleneck (much below peak)
- Datasets outstrip cache

<table>
<thead>
<tr>
<th>Machine</th>
<th>Peak (MF/s)</th>
<th>Triad (MB/s)</th>
<th>MF/MW</th>
<th>Eq. MF/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matt’s Laptop</td>
<td>1700</td>
<td>1122.4</td>
<td>12.1</td>
<td>93.5 (5.5%)</td>
</tr>
<tr>
<td>Intel Core2 Quad</td>
<td>38400</td>
<td>5312.0</td>
<td>57.8</td>
<td>442.7 (1.2%)</td>
</tr>
<tr>
<td>Tesla 1060C</td>
<td>984000</td>
<td>102000.0*</td>
<td>77.2</td>
<td>85000.0 (0.8%)</td>
</tr>
</tbody>
</table>

**Table:** Bandwidth limited machine performance

http://www.cs.virginia.edu/stream/
What are the Problems?

**Analysis of Sparse Matvec (SpMV)**

**Assumptions**
- No cache misses
- No waits on memory references

**Notation**
- $m$ Number of matrix rows
- $nz$ Number of nonzero matrix elements
- $V$ Number of vectors to multiply

We can look at bandwidth needed for peak performance

$$\left(8 + \frac{2}{V}\right) \frac{m}{nz} + \frac{6}{V} \text{ byte/flop}$$  \hspace{1cm} (2)

or achieveable performance given a bandwith $BW$

$$\frac{Vnz}{(8V + 2)m + 6nz} BW \text{ Mflop/s}$$  \hspace{1cm} (3)

Towards Realistic Performance Bounds for Implicit CFD Codes, Gropp, Kaushik, Keyes, and Smith.
What are the Problems?

Bandwidth

Improving Serial Performance

For a single matvec with 3D FD Poisson, Matt’s laptop can achieve at most

\[
\frac{1}{(8 + 2) \frac{1}{7} + 6} \text{bytes/flop}(1122.4 \text{ MB/s}) = 151 \text{ MFlops/s},
\]

which is a dismal 8.8% of peak.

Can improve performance by

- Blocking
- Multiple vectors

but operation issue limitations take over.
Improving Serial Performance

For a single matvec with 3D FD Poisson, Matt’s laptop can achieve at most

\[
\frac{1}{(8 + 2) \frac{1}{y} + 6} \text{ bytes/flop}(1122.4 \text{ MB/s}) = 151 \text{ MFlops/s}, \quad (4)
\]

which is a dismal 8.8% of peak.

Better approaches:

- Unassembled operator application (Spectral elements, FMM)
  - $N$ data, $N^2$ computation
- Nonlinear evaluation (Picard, FAS, Exact Polynomial Solvers)
  - $N$ data, $N^k$ computation
What are the Problems?

- Bandwidth
- Synchronization
- Convergence
Synchronization penalties can come from

- Reductions
  - GMRES orthogonalization
  - More than 20% penalty for PFLOTRAN on Cray XT5

- Small subproblems
  - Multigrid coarse problem
  - Lower levels of Fast Multipole Method tree
2 What are the Problems?
  - Bandwidth
  - Synchronization
  - Convergence
Convergence of the BFBT solve depends on
- Viscosity contrast (slightly)
- Viscosity topology
- Mesh

Convergence of the AMG Poisson solve depends on
- Mesh
Outline

1. 5 Slide Talk
2. What are the Problems?
3. Can we do Better?
   - BEM Formulation
   - BEM Solver
   - Interpolation
4. Advantages and Disadvantages
5. What is Next?
Use a **Boundary Element Method**, for the Laplace solves in BFBT, accelerated by **FMM**.
Missing Pieces

- BEM discretization and assembly
- Solver for BEM system
- Interpolation between FEM and BEM
Can we do Better?

- BEM Formulation
- BEM Solver
- Interpolation
The Poisson problem

\[ \Delta u(x) = f(x) \quad \text{on } \Omega \]  
\[ u(x) \big|_{\partial \Omega} = g(x) \]
The Poisson problem (Boundary Integral Equation formulation)

\[ C(\mathbf{x})u(\mathbf{x}) = \int_{\partial \Omega} F(\mathbf{x}, \mathbf{y})g(\mathbf{y}) - G(\mathbf{x}, \mathbf{y}) \frac{\partial u(\mathbf{y})}{\partial n} \, dS(\mathbf{y}) \]  

(5)

\[ G(\mathbf{x}, \mathbf{y}) = -\frac{1}{2\pi} \log r \]  

(6)

\[ F(\mathbf{x}, \mathbf{y}) = \frac{1}{2\pi r} \frac{\partial r}{\partial n} \]  

(7)
Restricting to the boundary, we see that

\[
\frac{1}{2} g(x) = \int_{\partial \Omega} F(x, y) g(y) - G(x, y) \frac{\partial u(y)}{\partial n} dS(y)
\]  

(5)
Discretizing, we have

\[-Gq = \left( \frac{1}{2} I - F \right) g\]  \hspace{1cm} (5)
Boundary Element Method

Now we can evaluate $u$ in the interior

$$u(x) = \int_{\partial \Omega} F(x, y)g(y) - G(x, y)\frac{\partial u(y)}{\partial n} dS(y)$$

(5)
Or in discrete form

\[ u = Fg - Gq \]  

(5)
The sources in the interior may be added in using superposition

\[
\frac{1}{2} g(x) = \int_{\partial \Omega} \left[ F(x, y) g(y) - G(x, y) \left( \frac{\partial u(y)}{\partial n} - f \right) \right] dS(y)
\] (5)
Can we do Better?

- BEM Formulation
- BEM Solver
- Interpolation
The solve has two pieces:

- **Operator application**
  - Boundary solve
  - Interior evaluation
  - Accomplished using the Fast Multipole Method

- **Iterative solver**
  - Usually GMRES
  - We use PETSc
Using the Fast Multiple Method, the Green's functions ($F$ and $G$) can be applied:

- in $O(N)$ time
- using small memory bandwidth
- in the interior and on the boundary
- with much higher serial and parallel performance
Fast Multipole Method

FMM accelerates the calculation of the function:

\[ \Phi(x_i) = \sum_j K(x_i, x_j)q(x_j) \]  

(6)

- Accelerates \( O(N^2) \) to \( O(N) \) time
- The kernel \( K(x_i, x_j) \) must decay quickly from \((x_i, x_i)\)
  - Can be singular on the diagonal (Calderón-Zygmund operator)
- Discovered by Leslie Greengard and Vladimir Rohklin in 1987
- Very similar to recent wavelet techniques
Fast Multipole Method

FMM accelerates the calculation of the function:

$$\Phi(x_i) = \sum_j \frac{q_j}{|x_i - x_j|}$$  \hspace{1cm} (6)

- Accelerates $O(N^2)$ to $O(N)$ time
- The kernel $K(x_i, x_j)$ must decay quickly from $(x_i, x_i)$
  - Can be singular on the diagonal (Calderón-Zygmund operator)
- Discovered by Leslie Greengard and Vladimir Rohklin in 1987
- Very similar to recent wavelet techniques
PetFMM CPU Performance

Strong Scaling

Can we do Better?

BEM Solver

ME Initialization
Upward Sweep
Downward Sweep
Evaluation
Load balancing stage
Total time

Time [sec]

Number of processors

ME Initialization
Upward Sweep
Downward Sweep
Evaluation
Load balancing stage
Total time

2 4 8 16 32 64 128 256

M. Knepley (UC)
Can we do Better?

PetFMM Load Balance

- uniform 4ML8R5
- uniform 10ML9R5
- spiral 1ML8R5
- spiral w/space-filling 1ML8R5
GPU Performance

- In our C++ code on a CPU, M2L transforms take 85% of the time
  - This does vary depending on $N$

- New M2L design was implemented using PyCUDA
  - Port to C++ is underway

- We can now achieve 500 GF on the NVIDIA Tesla
  - Previous best performance we found was 100 GF

- We will release PetFMM-GPU in the new year
GPU Performance

- In our C++ code on a CPU, M2L transforms take 85% of the time
  - This does vary depending on $N$

- New M2L design was implemented using PyCUDA
  - Port to C++ is underway

- We can now achieve 500 GF on the NVIDIA Tesla
  - Previous best performance we found was 100 GF

- We will release PetFMM-GPU in the new year
In our C++ code on a CPU, M2L transforms take 85% of the time. This does vary depending on $N$.

New M2L design was implemented using PyCUDA.
- Port to C++ is underway.

We can now achieve 500 GF on the NVIDIA Tesla.
- Previous best performance we found was 100 GF.

We will release PetFMM-GPU in the new year.
In our C++ code on a CPU, M2L transforms take 85% of the time
- This does vary depending on \( N \)

New M2L design was implemented using PyCUDA
- Port to C++ is underway

We can now achieve 500 GF on the NVIDIA Tesla
- Previous best performance we found was 100 GF

We will release PetFMM-GPU in the new year
PetFMM is an freely available implementation of the Fast Multipole Method
http://barbagroup.bu.edu/Barba_group/PetFMM.html

- Leverages PETSc
  - Same open source license
  - Uses Sieve for parallelism
- Extensible design in C++
  - Templated over the kernel
  - Templated over traversal for evaluation
- MPI implementation
  - Novel parallel strategy for anisotropic/sparse particle distributions
  - PetFMM—A dynamically load-balancing parallel fast multipole library
  - 86% efficient strong scaling on 64 procs
- Example application using the Vortex Method for fluids
- (coming soon) GPU implementation
Convergence

BEM Laplace operator is well-conditioned

- $\kappa = O(N_B) = O(\sqrt{N})$
  - Dijkstra and Mattheij

Thus the total work is in $O(N_B^2) = O(N)$

- Same as MG

Regular integral operators require only two multigrid cycles

- Multigrid of the 2nd kind by Hackbush
Can we do Better?

- BEM Formulation
- BEM Solver
- Interpolation
FEM $\rightarrow$ BEM
- FEM boundary conditions can be directly used in BEM
- May require a VecScatter

FEM $\leftarrow$ BEM
- BEM can evaluate the field at any domain point
- Cost is linear in the number of evaluations using FMM
- Can accommodate both
  - pointwise values, and
  - moments by quadrature
Outline

1. 5 Slide Talk
2. What are the Problems?
3. Can we do Better?
4. Advantages and Disadvantages
   - Bandwidth
   - Convergence
5. What is Next?
4 Advantages and Disadvantages
- Bandwidth
- Convergence
Advantages and Disadvantages

Bandwidth

Bandwidth and Serial Performance

- Provably low bandwidth
  - Shang-Hua Teng, SISC, 19(2), 635–656, 1998

- Key advantage over algebraic methods like FFT
  - Similar to wavelet transform

- Amenable to GPU implementation
  - Also highly concurrent
Advantages and Disadvantages

- Bandwidth
- Convergence
Advantages and Disadvantages

Convergence and Synchronization

- BEM matrices are better conditioned
  - However, FEM has better preconditioners
  - Without better preconditioners, might see more synchronization
  - Underexplored

- FMM can avoid bottleneck at lower levels
  - Overlap direct work with lower tree levels
  - Can provably eliminate bottleneck
Debatable Advantages

- Small memory
  - FEM can be done matrix-free

- Opens door to using Stokes operator for PC
  - We currently do not know what to do here
Outline

1. 5 Slide Talk
2. What are the Problems?
3. Can we do Better?
4. Advantages and Disadvantages
5. What is Next?
What is Next?

Direct Fast Method for Variable-Viscosity Stokes

- Complexity not currently precisely quantified
  - We would like a given number of flops/digit of accuracy

- Brute Force
  - Use BEM to compute layers between regions of constant viscosity
  - Better conditioned, but not direct

- Elegant method should be possible
  - The operator is pseudo-differential
  - “Kernel-independent” FMM exists
Direct Fast Method for Variable-Viscosity Stokes

- Complexity not currently precisely quantified
  - We would like a given number of flops/digit of accuracy

- Brute Force
  - Use BEM to compute layers between regions of constant viscosity
  - Better conditioned, but not direct

- Elegant method should be possible
  - The operator is pseudo-differential
  - “Kernel-independent” FMM exists
Direct Fast Method for Variable-Viscosity Stokes

- Complexity not currently precisely quantified
  - We would like a given number of flops/digit of accuracy

- Brute Force
  - Use BEM to compute layers between regions of constant viscosity
  - Better conditioned, but not direct

- Elegant method should be possible
  - The operator is pseudo-differential
  - “Kernel-independent” FMM exists