Stokes Preconditioning on a GPU

Matthew Knepley[1,2], Dave A. Yuen, and Dave A. May

[1]Computation Institute
University of Chicago

[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

BFBT

BFBT preconditions the Schur complement using

$$S_b^{-1} = L_p^{-1} G^T K G L_p^{-1} \tag{1}$$

where L_p is the Laplacian in the pressure space.

Current Problems

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

Alternative Proposal

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

Problems

The current BFBT code is limited by

- Bandwidth constraints
- Synchronization
- Convergence

Outline

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): $\mathbf{w} = \mathbf{y} + \alpha \mathbf{x}$
- Measures the memory bandwidth bottleneck (much below peak)
- Datasets outstrip cache

Machine	Peak (MF/s)	Triad (MB/s)	MF/MW	Eq. MF/s
Matt's Laptop	1700	1122.4	12.1	93.5 (5.5%)
Intel Core2 Quad	38400	5312.0	57.8	442.7 (1.2%)
Tesla 1060C	984000	102000.0*	77.2	8500.0 (0.8%)

Table: Bandwidth limited machine performance

http://www.cs.virginia.edu/stream/

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

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

or achieveable performance given a bandwith BW

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

Towards Realistic Performance Bounds for Implicit CFD Codes, Gropp, Kaushik, Keyes, and Smith.

Improving Serial Performance

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

$$\frac{1}{(8+2)\,\frac{1}{7}+6}\ \mathrm{bytes/flop}(1122.4\ \mathrm{MB/s}) = 151\ \mathrm{MFlops/s}, \tag{4}$$

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

$$\frac{1}{(8+2)\,\frac{1}{7}+6}\ \mathrm{bytes/flop}(1122.4\ \mathrm{MB/s}) = 151\ \mathrm{MFlops/s}, \qquad (4)$$

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

Outline

Synchronization

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

Outline

Convergence

Convergence of the BFBT solve depends on

- Viscosity constrast (slightly)
- Viscosity topology
- Mesh

Convergence of the AMG Poisson solve depends on

- Mesh

Outline

Alternative Proposal

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

Outline

Boundary Element Method

The Poisson problem

$$\Delta u(\mathbf{x}) = f(\mathbf{x}) \quad \text{on } \Omega \tag{5}$$

$$u(\mathbf{x})\,|_{\partial\Omega} = g(\mathbf{x}) \tag{6}$$

Boundary Element Method

The Poisson problem (Boundary Integral Equation formulation)

$$
\begin{aligned}
C(\mathbf{x})u(\mathbf{x}) &= \int_{\partial\Omega} F(\mathbf{x},\mathbf{y})g(\mathbf{y}) - G(\mathbf{x},\mathbf{y})\frac{\partial u(\mathbf{y})}{\partial n} dS(\mathbf{y}) && (5)\\
G(\mathbf{x},\mathbf{y}) &= -\frac{1}{2\pi}\log r && (6)\\
F(\mathbf{x},\mathbf{y}) &= \frac{1}{2\pi r}\frac{\partial r}{\partial n} && (7)
\end{aligned}
$$

Boundary Element Method

Restricting to the boundary, we see that

$$\frac{1}{2}g(\mathbf{x}) = \int_{\partial\Omega} F(\mathbf{x},\mathbf{y})g(\mathbf{y}) - G(\mathbf{x},\mathbf{y})\frac{\partial u(\mathbf{y})}{\partial n}dS(\mathbf{y}) \quad (5)$$

Boundary Element Method

Discretizing, we have

$$-Gq = \left(\frac{1}{2}I - F\right) g \tag{5}$$

Boundary Element Method

Now we can evaluate *u* in the interior

$$u(\mathbf{x}) = \int_{\partial\Omega} F(\mathbf{x},\mathbf{y})g(\mathbf{y}) - G(\mathbf{x},\mathbf{y})\frac{\partial u(\mathbf{y})}{\partial n} dS(\mathbf{y}) \tag{5}$$

Boundary Element Method

Or in discrete form

$$u = Fg - Gq \tag{5}$$

Boundary Element Method

The sources in the interior may be added in using superposition

$$\frac{1}{2}g(\mathbf{x}) = \int_{\partial\Omega} F(\mathbf{x},\mathbf{y})g(\mathbf{y}) - G(\mathbf{x},\mathbf{y})\left(\frac{\partial u(\mathbf{y})}{\partial n} - f\right) dS(\mathbf{y}) \tag{5}$$

Outline

BEM Solver

The solve has two pieces:

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

- Iterative solver
  - Usually GMRES
  - We use PETSc

Operator Application

Using the Fast Multiple Method,
the Green's functions (F and G) can be applied:

- in $\mathcal{O}(N)$ 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:

$$\Phi(x_i) = \sum_j K(x_i, x_j) q(x_j) \tag{6}$$

- Accelerates $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$ 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:

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

- Accelerates $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$ 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