Parallel FMM

Matthew Knepley

Computation Institute
University of Chicago
Department of Molecular Biology and Physiology
Rush University Medical Center

Conference on High Performance Scientific Computing
In Honor of Ahmed Sameh’s 70th Birthday
Purdue University, October 11, 2010
Using estimates and proofs, a simple software architecture gets good scaling, efficiency, and adaptive load balance.
Using estimates and proofs, a simple software architecture, gets good scaling, efficiency, and adaptive load balance.
Using estimates and proofs, a simple software architecture gets good scaling, efficiency, and adaptive load balance.
The **PetFMM team:**

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

- **Dr. Felipe Cruz**, developer of GPU extension
  - Nagasaki Advanced Computing Center, Nagasaki University

- **Dr. Rio Yokota**, developer of 3D extension
  - Dept. of Mechanical Engineering, Boston University
Collaborators

Chicago Automated Scientific Computing Group:

- **Prof. Ridgway Scott**
  - Dept. of Computer Science, University of Chicago
  - Dept. of Mathematics, University of Chicago

- **Peter Brune**, (biological DFT)
  - Dept. of Computer Science, University of Chicago

- **Dr. Andy Terrel**, (Rheagen)
  - Dept. of Computer Science and TACC, University of Texas at Austin
Complementary Work

FMM Work

- Queue-based hybrid execution
  - OpenMP for multicore processors
  - CUDA for GPUs

- Adaptive hybrid Treecode-FMM
  - Treecode competitive only for very low accuracy
  - Very high flop rates for treecode M2P operation

- Computation/Communication Overlap FMM
  - Provably scalable formulation
  - Overlap P2P with M2L
FMM Applications

FMM can accelerate both integral and boundary element methods for:

- Laplace
- Stokes
- Elasticity
FMM Applications

FMM can accelerate both integral and boundary element methods for:
- Laplace
- Stokes
- Elasticity

Advantages
- Mesh-free
- $O(N)$ time
- Distributed and multicore (GPU) parallelism
- Small memory bandwidth requirement
FMM accelerates the calculation of the function:

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

- 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
FMM accelerates the calculation of the function:

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

- 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
Pairs of boxes are divided into *near* and *far*:
Pairs of boxes are divided into *near* and *far*: 

Neighbors are treated as *very near*. 
Short Introduction to FMM

Functional Decomposition

- Create Multipole Expansions.
- Evaluate Local Expansions.

- P2M
- M2M
- M2L
- L2L
- L2P

Downward Sweep

Upward Sweep
FMM in Sieve

- The Quadtree is a Sieve with optimized operations
  - Multipoles are stored in Sections
  - Two Overlaps are defined
    - Neighbors
    - Interaction List
  - Completion moves data for
    - Neighbors
    - Interaction List
The Quadtree is a Sieve with optimized operations. Multipoles are stored in Sections. Two Overlaps are defined: Neighbors, Interaction List. Completion moves data for Neighbors, Interaction List.
FMM in Sieve

- The Quadtree is a *Sieve* with optimized operations
- Multipoles are stored in *Sections*
- Two Overlaps are defined
  - Neighbors
  - Interaction List
- Completion moves data for
  - Neighbors
  - Interaction List
FMM in Sieve

- The Quadtrees is a Sieve
  - with optimized operations
- Multipoles are stored in Sections
- Two Overlaps are defined
  - Neighbors
  - Interaction List
- Completion moves data for
  - Neighbors
  - Interaction List
The Quadtree is a **Sieve**
- with optimized operations
- **Multipoles are stored in** Sections
- **Two Overlaps are defined**
  - Neighbors
  - Interaction List
- Completion moves data for
  - Neighbors
  - Interaction List
The Quadtrees is a Sieve with optimized operations. Multipoles are stored in Sections. Two Overlaps are defined for Neighbors and Interaction List. Completion moves data for Neighbors and Interaction List.
The Quadtree is a **Sieve** with optimized operations.

- **Multipoles** are stored in **Sections**
- **Two Overlaps** are defined:
  - Neighbors
  - Interaction List
- **Completion moves data for**:
  - Neighbors
  - Interaction List
Parallelism

FMM Control Flow

Upward Sweep

Downward Sweep

Create Multipole Expansions. Evaluate Local Expansions.
P2M M2M M2L L2L L2P

Kernel operations will map to GPU tasks.
Kernel operations will map to GPU tasks.
Parallel Tree Implementation

- Divide tree into a root and local trees
- Distribute local trees among processes
- Provide communication pattern for local sections (overlap)
  - Both neighbor and interaction list overlaps
  - Sieve generates MPI from high level description
Parallel Tree Implementation

How should we distribute trees?

- Multiple local trees per process allows good load balance
- Partition weighted graph
  - Minimize load imbalance and communication
  - Computation estimate:
    - Leaf: \( N_i p (P2M) + n_i p^2 (M2L) + N_i p (L2P) + 3^d N_i^2 (P2P) \)
    - Interior: \( n_c p^2 (M2M) + n_i p^2 (M2L) + n_c p^2 (L2L) \)
- Communication estimate:
  - Diagonal: \( n_c (L - k - 1) \)
  - Lateral: \( 2^d \frac{2^m (L - k - 1) - 1}{2^m - 1} \) for incidence dimension \( m \)
- Leverage existing work on graph partitioning
  - ParMetis

- Good partitions exist for non-uniform distributions
  - \( O \left( \sqrt{n} (\log n)^{3/2} \right) \) edgecut
  - \( O \left( n^{2/3} (\log n)^{4/3} \right) \) edgecut

- As scalable as regular grids

- As efficient as uniform distributions

- ParMetis will find a nearly optimal partition

- Good partitions exist for non-uniform distributions
  - 2D \( C_i = 1.24^i C_0 \) for random matching
  - 3D \( C_i = 1.21^i C_0 \) for random matching

- 3D proof needs assurance that average degree does not increase

- Efficient in practice
Parallel Tree Implementation

Advantages

- Simplicity
  - Complete serial code reuse
  - Provably good performance and scalability
Parallel Tree Implementation

Advantages

- Simplicity
- Complete serial code reuse
- Provably good performance and scalability
Parallelism

Parallel Tree Implementation

Advantages

- Simplicity
- Complete serial code reuse
- Provably good performance and scalability
Distributing Local Trees

The interaction of local trees is represented by a weighted graph.

This graph is partitioned, and trees assigned to processes.