Lecture 21:
Tree codes

David Bindel

12 Apr 2010

Logistics

- April 19, SCAN seminar: Padma Raghavan
- April 21, guest lecture: Charlie Van Loan

Recurring theme

Lots of problems can be partitioned as

- A "coarse" problem that captures long-range effects
- A "fine" problem that handles local effects

So far: multigrid, multilevel graph partitioning. Today: tree codes.

General algebraic picture

Several problems reduce to one of these:

$$\phi(\mathbf{x}) = \int_\Gamma K(\mathbf{x}, \mathbf{y}) w(\mathbf{y})\, d\mathbf{y}$$

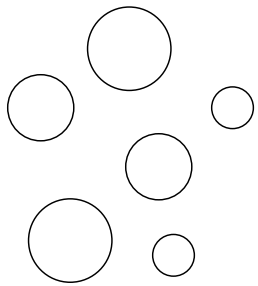
$$\phi(\mathbf{x}) = \sum_{i=1}^{N} K(\mathbf{x}, \mathbf{y}_i) w_i$$

where the *kernel* $K(\mathbf{x}, \mathbf{y})$ is "nice."

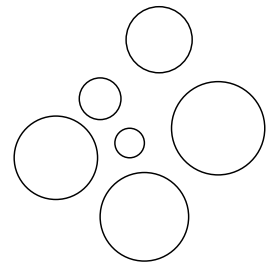
- Force evaluations in N-body codes
- Integral equations (Laplace, Helmholtz, etc)
  - Stokes flows in fluid dynamics
  - Electrostatic fields (e.g. for capacitance)
  - Coulumbic interaction in DFT codes

We want to compute this fast.

Example: Gravitational interaction



A



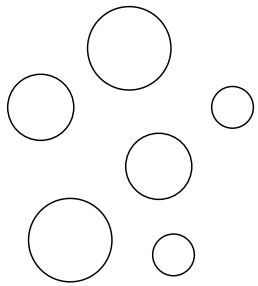
B

Gravitational potential from N bodies (in 3D): $F(\mathbf{x}) = -\nabla\phi(\mathbf{x})$,

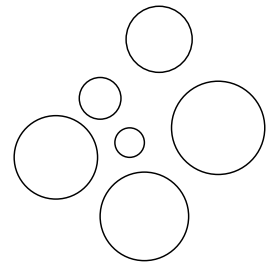
$$\phi(\mathbf{x}) = -\sum_{i=1}^{N} \frac{m_i}{\|\mathbf{x} - \mathbf{y}_i\|}, \qquad \mathbf{F} = -\sum_{i=1}^{N} \frac{\mathbf{x} - \mathbf{y}_i}{\|\mathbf{x} - \mathbf{y}_i\|^3} m_i.$$

If $\mathbf{x} - \mathbf{y}_i$ and $\mathbf{x} - \mathbf{y}_j$ are "close", then $\|\mathbf{x} - \mathbf{y}_i\|^{-1} \approx \|\mathbf{x} - \mathbf{y}_j\|^{-1}$.

Example: Gravitational interaction



A



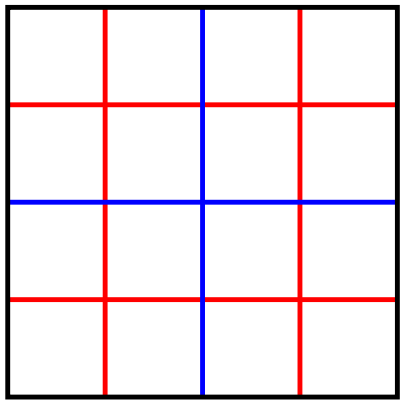
B

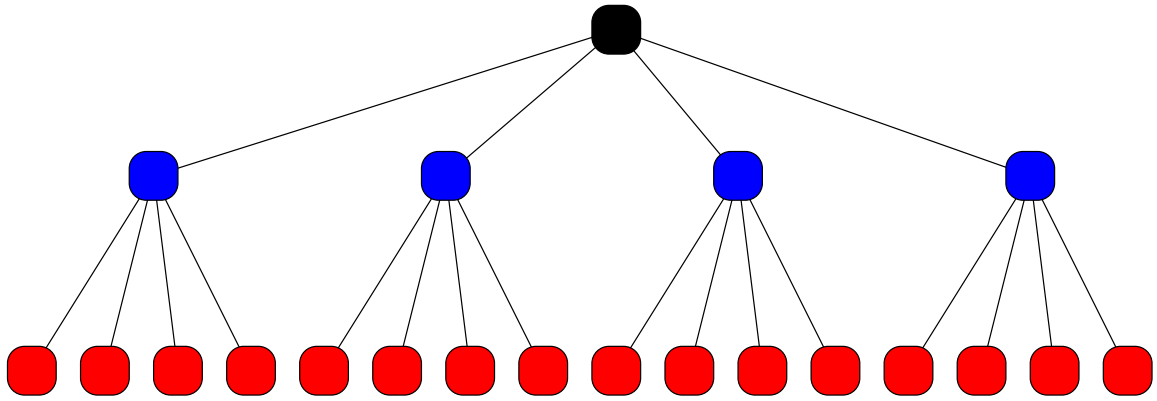
Cluster points: $\{\mathbf{y}_{j,i}\}$ for ith particle in cluster j, $\bar{\mathbf{y}}_j$ for centroid

$$\phi(x) = -\sum_{j=0}^{N}\sum_{i=1}^{N_j}\frac{m_{j,i}}{\|\mathbf{x}-\mathbf{y}_{j,i}\|} \approx -\sum_{i=1}^{N_0}\frac{m_{0,i}}{\|\mathbf{x}-\mathbf{y}_i\|} - \sum_{j=1}^{K}\frac{M_j}{\|x-\bar{\mathbf{y}}_j\|}$$

where M_j is the total mass in cluster j. How to build good clusters?

Quadtrees and octree





Basic idea: partition space with a quadtree or octtree

- Recursively cut boxes in four (2D) or eight (2D)
- Subdivision become children for parent box
- Can be adaptive (subdivide some children, not others)
- Assign each box a total mass, centroid
- Interact x with small boxes nearby, large boxes far off

Adaptive quadtree construction

```
# insert particle j at node n
function quadtree_insert(j,n)
  if n has children
    determine child c to which j belongs
    quadtree_insert(j,c)
  elseif n contains a particle p
    add four children of n
    move p to the appropriate child
    determine child c to which j belongs
    quadtree_insert(j,c)
  else
    store particle j at n
  end
```

Cost = $O(Nb)$, b is number of bits in particle coordinates.
Uniform case: $O(N \log N)$

Center of mass

```
# Decorate node n with a mass M and center x
function quadtree_mass(n)
  if n has children
    M = 0
    x = 0
    for each child c
      quadtree_mass(c)
      M += mass of c
      x += center of c * mass of c
    x /= M
  else
    M = mass of particle at n
    x = position of particle at n
```

Cost is number of nodes in quadtree ($O(N)$ in uniform case).

Force computation (Barnes-Hut)

```
# Compute force from particles in node n
# on a test mass at position x
function force(n,x)
  y = center of mass of n
  D = size of n
  if D < theta*|x-y|
    compute force using centroid approximation
  else
    f = 0
    for each child c
      f += force(c,x)
```

Cost is $O(N \log N)$, potentially large constant.

Barnes-Hut summary

"A hierarchical $O(n \log n)$ force calculation algorithm", J. Barnes and P. Hut, Nature, 324 (1986)... and many others

- Build adaptive quad/octtree
- Compute center of mass and total mass per node
- Use center-of-mass approximation when accurate enough

This gets expensive at high accuracy requirements...

Beyond Barnes-Hut

Barnes-Hut uses a simple center-of-mass approximation.
What if we use more about the particle distribution?

Fast Multipole Method

"A fast algorithm for particle simulation", L. Greengard and V. Rokhlin, J. Comp. Phys., 73 (1987)... and many later papers.

Basic idea:

- Upward pass: compute outer expansions for far field
- Downward pass: compute inner expansions of far field

Use *multipole* for outer expansion, *Taylor* for inner.
Use *translation operators* to re-center expansions.

Example: 2D electrostatic

Potential and force from particle at origin:

$$\phi(\mathbf{r}) = \log \|\mathbf{r}\|$$
$$\mathbf{F} = -\mathbf{r}/\|\mathbf{r}\|^2$$

Think $z = x + iy$. Then

$$\phi(z) = \log |z| = \Re(\log z)$$
$$\mathbf{F} = -z/|z|^2$$

Let's just keep complex potential.

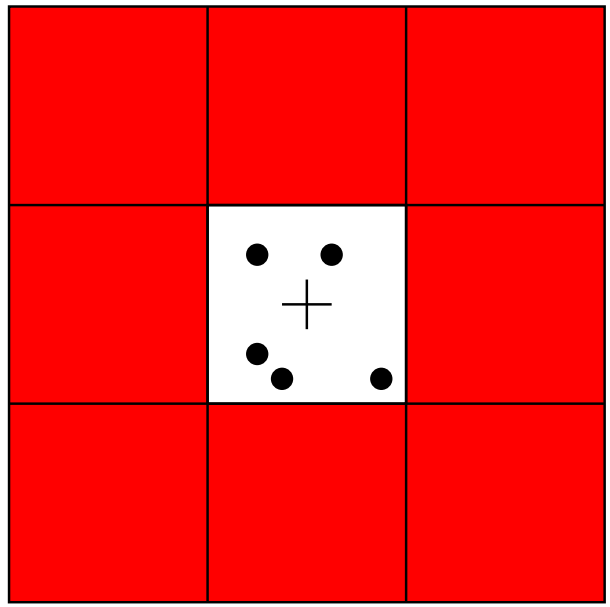
Multipole expansion

Basic expansion:

$$\phi(z) = \sum_{k=1}^{n} m_k \log(z - z_k) = M \log(z) + \sum_{j=1}^{\infty} \alpha_j z^{-j}$$
$$= M \log(z) + \sum_{j=1}^{r} \alpha_j z^{-j} + O\left(\left[\max_k \frac{|z_k|}{|z|} \right]^{r+1} \right).$$

If points lie in a D-by-D box at origin, than for any point outside a $3D$-by-$3D$ box, error is less than 2^{-r+1}

Multipole expansion



If points lie in a D-by-D box at origin, than for any point outside a $3D$-by-$3D$ box, error is less than 2^{-r+1}

Translation

- *Translation* operators re-center multipole expansion.
- Can implement as simple matvec on expansion coefficients.
- Use to combine several multipole expansions.
- Build expansions on quadtree as with center-of-mass.

Conversion

- Can *convert* multipole to Taylor for a box
- For each node, local potential is:
  1. Contributions from nearest neighbor boxes at same level
  2. Contributions from nearest neighbors at parent level
  3. Contributions from things further away
- Last two can be computed via outer-to-inner conversion and translation of inner expansions
- And recurse!
- ... using direct evaluation at bottom level.

Total cost: $O(N)$

Kernel-independent FMM

- Takes some coding work to build the expansions, translations!
- But kernel-dependent head-scratching goes into:
  - Computation of the expansions
  - Translating expansions
- Approximate outside potential by point distribution on bounding circle
- Distribution becomes expansion
- Build automatic ways to compute, translate distribution for smooth enough kernels

See KiFMM3D code of Harper Langston.

Parallel implementation

"A massively parallel adaptive fast-multipole method on heterogeneous architectures" Lashuk et al, SC 09.

- Uses kernel-independent FMM for basic algorithm
- Massive MPI code at the top level
- GPU accelerators at the bottom level

Work partitioning

Partition space using the octtree:

- Enumerate leaves in space-filling curve order
- Use sampling to estimate work for different splits

Locally Essential Tree

"A parallel hashed oct-tree N-body algorithm", M. Warren and J. Salmon, SC 93.

- Partition work across leaf octants
- Take one processors's leaf octants
- LET == part of quadtree needed to evaluate those octants
- Sender decides what other processors need its parts, then sends these subsets (via position in quadtree)
- Will set up "ghost" octants to receive neighbor data

Parallel software + reading material

- Papers to get started
  - Original Greengard-Rokhlin paper is quite readable
  - "A short course on fast multipole methods", Beatson and Greengard
- Software
  - KiFMM3D (Langston)
  - PetFMM (Cruz, Knepley, Barba)

... and all sorts of other packages (some less accessible).

Hierarchical matrices and FMM

FMM says far-range interactions can be summarized via a few intermediate variables. Algebraically, this yields matrices with interesting low-rank structure.

- $\mathcal{H}$-matrices (Hackbush)
- Semi-separable matrices (see book of Vandebril, Van Barel, Mastronardi)
- Skeletonization and related ideas (see 2009 Acta paper of Greengard, Gueyffier, Martinsson, Rokhlin)
- Freely available parallel solver software?