Parallel Implementation of multipole-based Poisson-Boltzmann solver

Eng Hui Yap
CS 267 Project
May 11, 2009
Simulation Overview

**Implicit Solvent**
\[ \varepsilon_s = 78, \kappa > 0 \]

**Protein(s):**
\[ \varepsilon_p = 4, \kappa = 0 \]

1. **Initialize system**
2. **Calculate forces**
   - Solve linearized Poisson Boltzmann Equation (LPBE)
   \[
   - \nabla [\varepsilon(r) \nabla \Phi(r)] + \kappa^2 \Phi(r) = \rho_{fixed}(r)
   \]
3. **Propagate Molecules**
   - Brownian Dynamics using forces from (2)
4. **Repeat** 2-3 until criteria is met
Each molecule is represented as a collection of spheres.

For each sphere $k_i$:
1. Calculate surface charge multipole $S_{nm}$
   (i) Express $\Phi_{in}$ and $\Phi_{out}$ in terms of multipoles
   (ii) Setting up boundary equations.
   (iii) Solve for $S_{nm}$
2. Update contribution from $S_{nm}$ to other spheres
3. Repeat for all spheres until convergence criteria is reached
(i) Potential Equations (in terms of multipoles)

**Inside sphere ki:**

\[
\Phi_{in}^{(ki)}(\mathbf{r}) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \left( \frac{E_{Fixed \, nm}^{(ki)}}{r^{n+1}} + r^{n} B_{nm}^{(ki)} \right) Y_{nm}(\theta, \phi)
\]

**Outside sphere ki:**

\[
\Phi_{out}^{(ki)}(\mathbf{r}) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \left( \frac{(E_{Fixed} + S)_{nm}^{(ki)}}{r^{n+1}} + (L_s + L_E + L_{Ext})_{nm}^{(ki)} r^{n} \right) Y_{nm}(\theta, \phi)
\]

Goal: Solve for unknown S
(ii) Boundary conditions

On sphere \( ki \)'s surface \((a, \theta, \phi)\):

\[
\Phi_{\text{in}}(\mathbf{r})\bigg|_{\text{Surface}_{ki}} = \Phi_{\text{out}}(\mathbf{r})\bigg|_{\text{Surface}_{ki}}
\]

\[
\varepsilon_{\text{in}} \frac{d\Phi_{\text{in}}(\mathbf{r})}{dn}\bigg|_{\text{Surface}_{ki}} = \varepsilon_{\text{out}}(\theta, \phi) \frac{d\Phi_{\text{out}}(\mathbf{r})}{dn}\bigg|_{\text{Surface}_{ki}}
\]

\[
\sum_{n=0}^{\infty} \sum_{m=-n}^{n} \left[ n\varepsilon_{p} + (n+1)\varepsilon_{\text{out}}(\theta, \phi) \right] S_{nm}^{(ki)} \ Y_{nm}(\theta, \phi)
\]

\[
= \left( \varepsilon_{\text{out}}(\theta, \phi) - \varepsilon_{p} \right) \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \left\{ -(n+1)E_{nm}^{(ki)} + an(L_s + L_E + L_{Ext})_{nm}^{(ki)} \right\} Y_{nm}(\theta, \phi)
\]

\[
\sum_{n=0}^{\infty} \sum_{m=-n}^{n} X_{nm}^{(ki)} Y_{nm}(\theta, \phi)
\]

\[
(*)
\]
(iii) Solving Boundary Equation (*) for $S_{nm}$

Represent (*) as linear system of equations, solve $S_{nm}$ up to $p$ poles:

Method 1: Linear Least Square (LLS) solvers

\[(n,m) \rightarrow S_{nm} \quad \text{RHS}_{(\theta,\phi)} \]

Requires LLS solver
-> Inefficient!
For $p=60$: ~ 10min per solution

Method 2: Analytical, iterative method using orthonormality property of SH

\[(l,s) \rightarrow X'_{ls} \quad \text{Imat} \quad \rightarrow \quad S_{nm} \]

Matrix-Vector Multiply
-> Fast
For $p=60$:
Initial matrix prep ~ 14min per sphere
Subsequent solution ~ 0.4s
Simulation Algorithm (Serial)

Initialization
- For each sphere:
  - Calculate Surface Integrals
  - Compute polarization matrix \((Imat)\)

Production Run
- For each sphere:
  - Update contributions from other spheres
- Solve till all \(S_{nm}\) converges
- Calculate desired quantities (potential, forces, etc)
- Move proteins

Docked?
- No
- Yes END
**Parallization Strategy**

**Parallelization at sphere level**
- solve $S_{nm}$ for each sphere separately and share updated values with other spheres
- Jacobi iteration vs. Gauss-Seidel iterations

1) **Shared Memory Only Model**
- adequate for small systems ($< 10$ spheres)
  - Using OpenMP
  - Easy implementation within C++ object-oriented code

2) **Hybrid Model**
- required for larger scale systems ($> 10$ spheres)
  - Intra-node: shared memory using OpenMP
  - Inter-node: distributed memory using MPI
  - C++ objects need to be packed/unpacked for MPI communications
For each sphere (OMP):
- Calculate Surface Integrals
- Compute polarization matrix ($Imat$)

Update contributions from other spheres

Solve till all $Snm$ converges

Calculate desired quantities (potential, forces, etc)

Move proteins

Docked?

No

Yes

END
For each node (MPI):
  For each sphere (OMP):
    - Calculate Surface Integrals
    - Compute polarization matrix ($Imat$)

For each node (MPI):
  Update contributions from other spheres

For each sphere (OMP):
  Solve till all $S_{nm}$ converges

Calculate desired quantities (potential, forces, etc)

Move proteins

Docked?

Yes → END

No
Test cases for Timing

A) 8 overlapping spheres

- Different no. of poles used (p = 5, 10, 30, 60)
- Different no. of threads used (t = 1, 2, 4, 8)
Preliminary Timing Results (Shared Memory)

Time Per polarization cycle (8 spheres)

- Time [s]
- Number of Threads
- p=5
- p=10
- p=30
- p=60