Optimizing Espresso: Domain Decomposition for Constraints

Hendrick W. de Haan
David Sean

Physics Department, University of Ottawa, Ottawa, CA

MCD 215
Ottawa
December 10, 2009
**Introduction: MD Basics**

**MD Algorithm:**
- find net force on particle & numerically integrate

**Program Flow:**
- Loop over particles:
  - calculate net force:
    - Short Range Non-bonded interactions
    - Constraints
    - Bonded interactions
    - External Forces
    - Long Ranged Non-bonded interactions
  - numerically integrate ($a \rightarrow \Delta v \rightarrow \Delta x$) & update positions

**For many of our systems:**
- Short range non-bonded: many - WCA
- Constraints: many - WCA
- Bonded interactions: polymer only - FENE
- External forces: yes but trivial
- Long ranged non-bonded: none (!)

**Most Time spent:**
- Short-ranged non-bonded: on particle $i$, find distance to particle (object) $j$ and do forces

$$\sum F_i = m_i a_i$$

Different needs than the Germans!
**Domain Decomposition**
- break the system into domains
- particle only looks at particles in same or neighbouring cell
- \( t \sim N \)

**Generic System:**
- \( N \) particles in a box

**Brute Force: \( N^2 \)**
- each particle \( N \) checks other particles \( N \)
- \( t \sim N^2 \)

**Domain Decomposition**
- break the system into domains
- particle only looks at particles in same or neighbouring cell
- \( t \sim N \)
Obstacles:
- particles (objects) fixed in space:
  - define areas of excluded volume in the system
- Espresso does not have a special particle type for this
- two options:
  - fixed normal particles
  - constraints
The Problem with Obstacles

Fixed Particles:

- good: particles look at obstacles
- bad: obstacles look at particles/obstacle
- since DD, extra time is due to obst as part:
  \[ t_{\text{extra}} \sim N_{\text{obst}} \]
- always updating as if particle...

Constraints:

- good: constraints don’t loop over constraints
- bad: particles look at every constraint
- each particle checks \( N_{\text{con}} \)
  \[ t_{\text{extra}} \sim (N_{\text{con}})(N_{\text{part}}) \]
- smaller prefactor: only distance check
Translocation in Crowded Environment

- 2000 spherical particles:

\[ N_{\text{obst}} = N_{\text{con}} \]

- \( N_{\text{obst}} << N_{\text{con}}N_{\text{part}} \)

- do fixed particles

Polymer in a gel

- box 75x75x75 w/ con every 5

- 588 cylinders: 3x(14x14)

- 44,100 particles

\[ N_{\text{con}} << N_{\text{obst}} \]

- \( N_{\text{con}}N_{\text{part}} \sim 117,600 \)

- because not full particles, faster to do constraints
Either Method is a Rip-off

Obstacles
- particles (objects) fixed in space
- only polymer moving
- time should be almost independent of $N_{\text{obst}}$

Solutions:
1) New class of particles: obstacles
   - regular DD of system
   - particles see obstacles, not vice-versa
   Advantage:
   - eliminate calculations for fixed particles
   Trouble:
   - need (Newton = FALSE)
   - pain to code into Espresso

2) DD for constraints
   - regular DD of system
   - custom DD for constraints
   Advantage:
   - particles only look at close constraints
   Trouble:
   - code DD for constraints
New Approach

Program Flow
- Espresso does DD for particles-particles
  - don’t mess with this

- when calculating forces
  - in integration loop: verlet.c
  - call to add constraints: go to our code

- if first time, do DD for constraints

- regardless, do constraint forces
  - use DD-constraint information

Efficiency
- closer to $t \sim N_{\text{part}}$ scaling
- penalty is increased overhead: a one-time cost!
**Technical Details**

**DD constraints**
- 3D array of structures
  - carve space into cells in x,y,z
- structure has members:  
  - population
  - list of constraints in that cell
- fixed cell size: $2\sigma \times 2\sigma \times 2\sigma$
  - interaction $\sim \sigma$
  - since we look over neighbours, safe to catch all
  - means box length must be even!
- carve space:
  - if spheres: loop over constraints and place them in cell
  - if other: loop over space and if constraint in cell, add it

**Per particle**
- get DD-con index from particle x,y,z
- do constraint forces with list of constraints in the index cell and adjacent cells

**Files (Hack - will be redone by Dav clean!)**
- most work done in constraint.h (corresponding changes in constraint.c)
- altered calls in verlet.c/verlet.h
- minor changes in: some other files
Test System: Spheres

- 50x50x50 system

**Vanilla Espresso:**
- 4450 fixed particles
- 100 x 100 integration steps

**Frankenstein Espresso:**
- 4450 spherical constraints
- 1000 x 100 integration steps
Test System: Cylinders

- 100x100x100 system
- 325 cylinders

Vanilla Espresso
- 100 x 100 integration steps

Frankenstein Espresso
- 1000 x 100 integration steps
Efficiency Gains: Crowded

Translocation in Obstacles: Old System
- 50 monomers
- 1 nanopore constraint, **2000 obstacles**
- 100x100 MD
- about 1 week -> 1 day

<table>
<thead>
<tr>
<th></th>
<th>Vanilla - particles</th>
<th>Vanilla - constraints</th>
<th>Frankenstein</th>
</tr>
</thead>
<tbody>
<tr>
<td>box</td>
<td>25x25x50</td>
<td>25x25x50</td>
<td>50x50x50</td>
</tr>
<tr>
<td>time (MD)</td>
<td>16.84</td>
<td>102.7</td>
<td>2.41</td>
</tr>
<tr>
<td>t/t_{frank}</td>
<td>6.99</td>
<td>42.7</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Efficiency Gains: Crowded

Translocation in Obstacles: New Big System
- 50 monomers
- 1 nanopore constraint, **2662 obstacles**
- box is 200x200x200
- inner box is 25x25x50
- 100x100 EM, 100x100 EQ, 100x100 MD

- Vanilla doesn’t like boxes this big
- Frankenstein doesn’t care that much

<table>
<thead>
<tr>
<th></th>
<th>Vanilla - particles</th>
<th>Frankenst ein</th>
<th>V/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overhead</td>
<td>0.230</td>
<td>0.250</td>
<td>0.92</td>
</tr>
<tr>
<td>EM</td>
<td>102.08</td>
<td>2.26</td>
<td>45.22</td>
</tr>
<tr>
<td>EQ</td>
<td>102.02</td>
<td>2.15</td>
<td>47.50</td>
</tr>
<tr>
<td>MD</td>
<td>104.11</td>
<td>2.58</td>
<td>40.72</td>
</tr>
<tr>
<td>Total</td>
<td>30.84</td>
<td>7.21</td>
<td>42.77</td>
</tr>
</tbody>
</table>
Efficiency Gains: Crowded

Translocation in Obstacles:
- 1 nanopore constraint, $N$ spherical constraints
- 1000x100 integration steps per frame
- small box Vanilla, big box Frankenstein

<table>
<thead>
<tr>
<th>L</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>45</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>N obst</td>
<td>2,661</td>
<td>4,394</td>
<td>6,750</td>
<td>9,826</td>
<td>13,718</td>
<td>18,572</td>
</tr>
<tr>
<td>Vanilla T</td>
<td>242.2</td>
<td>519.2</td>
<td>1057.9</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Frank T</td>
<td>26.77</td>
<td>27.69</td>
<td>31.26</td>
<td>35.44</td>
<td>41.32</td>
<td>47.26</td>
</tr>
</tbody>
</table>

$T/L = \frac{0.133N}{N_{25}} + 0.839$
Efficiency Gains: Gel

Dav-like system:
- **588 constraints**: $3 \times (14 \times 14)$ cylinders
- 200 monomers
- 10 cycles x 100 integration steps EM
- 10 cycles x 100 integration steps MD
- time in seconds
Efficiency Gains: Gel

Short Run:
- 10 cycles x 100 integration steps EM
- 10 cycles x 100 integration steps MD
- time in seconds

Little Longer Run:
- 10 cycles x 100 integration steps EM
- 100 cycles x 100 integration steps MD
- time in seconds

Long Runs:
- for long runs, will be ~ 14 times faster
- 2 week runs are overnight now

<table>
<thead>
<tr>
<th></th>
<th>Vanilla</th>
<th>Franken-stein</th>
<th>V/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overhead</td>
<td>0.066</td>
<td>5.416</td>
<td>0.012</td>
</tr>
<tr>
<td>EM</td>
<td>25.269</td>
<td>1.785</td>
<td>14.156</td>
</tr>
<tr>
<td>MD</td>
<td>24.450</td>
<td>1.801</td>
<td>13.576</td>
</tr>
<tr>
<td>Total</td>
<td>49.785</td>
<td>9.002</td>
<td>5.53</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Vanilla</th>
<th>Franken-stein</th>
<th>V/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overhead</td>
<td>0.060</td>
<td>5.581</td>
<td>0.011</td>
</tr>
<tr>
<td>EM</td>
<td>22.502</td>
<td>1.896</td>
<td>11.868</td>
</tr>
<tr>
<td>MD</td>
<td>245.048</td>
<td>16.906</td>
<td>14.495</td>
</tr>
<tr>
<td>Total</td>
<td>267.61</td>
<td>24.383</td>
<td>10.975</td>
</tr>
</tbody>
</table>
Still bugs:
- code needs to be tested

Not clean implementation:
- modular: develop separate obstacle.h & obstacle.c
- minimize footprint on native Espresso
- Dav has good idea for clean implementation

Fixed constraints:
- anticipate we can extend to periodically updated obstacle positions easily
- will be some penalty to efficiency

Even number for box size:
- small price to pay

Not parallelized:
- difficult?
Advantages

Espresso:
- keep the capabilities, flexibility, and convenience of using Espresso
- optimized for Slater-esque systems

Shines for:
- systems with static, complicated geometry
- currently:
  - nanopore translocation in crowded environment
    - Hendrick
  - polymer mobility in a gel
    - David
    - Sami
  - nanofluidic device
    - Antoine

Near Future:
- incorporate Tyler’s SRD
- hydrodynamic interactions
- very efficient tool to do complicated geometry with flows: nanofluidics
Word
- wooooooooooooooord