Massively Parallel Algorithms for Large Scale, Long-Term Molecular Constrained Dynamics Simulations

Andrés Jaramillo-Botero
Pontificia Universidad Javeriana, Cali, Colombia
ajaramil@puj.edu.co or ajaramil@wag.caltech.edu

Computational Nanotechnology and Molecular Engineering Workshop
Materials and Process Simulation Center, Caltech, 2004
Outline

1. Motivation and Abstract
2. Molecular Dynamics (MD)
   - Fundamentals of Molecular Dynamics
   - State of the Art MD
   - Parallelization of MD Simulations
3. Constrained Rigid Body Dynamics: CADE
   - Problems: Isoefficiency and Scaling
   - The need for parallelizing the Equations of Motion
   - Serial Chain Rigid Multi-Bodies
   - Hyper-branched Systems
   - Closed Chain Systems
4. Implementation Issues and Results
5. Conclusions
6. Acknowledgments
General Motivation

OPEN PROBLEMS:
Size and time scales supported do NOT permit treatment of important problems:

- Dynamics of self-assembled nano-scale devices and systems.
- State transformations, such as: conformational transformations in proteins.
- Structure and properties of crystalline and membrane polymers.
- Structure and properties of biological systems, such as: globular proteins, biological membranes, DNA packing (nucleosides).

REQUIREMENT:

Significant Improvements in computational efficiency (orders of magnitude !!!).
Why Parallel Molecular Dynamics?

- MD is effective for a nano-scale phenomenon analysis
- Computation time is dominated by calculation of forces
  - direct summation algorithm for calculating long-range interactions is $O(N^2)$
  - cut-off method
- Need to simulate a system without a truncation of long-range forces
  - e.g. simulation of the restoration mechanism of damaged DNA
- Two main approaches:
  - to implement fast serial algorithms
    - Barnes-Hut tree method, Fast Multipole Method (FMM)
  - to implement fast parallel algorithms on parallel computers
    - Parallel FMM, Parallel CMM, Rigid Body Dynamics
Molecular Dynamics:

Ideal Calculations:

Solve the time-dependent Schrödinger equation for nuclei and electrons:

$$\left[ \frac{\hbar}{2m} \nabla^2 + V(R, t) \right] \psi(R, t) = i\hbar \frac{\delta \psi(R, t)}{\delta t}$$

$\hbar$ is the Planck constant over $\pi$ and $V(R, t)$ is the Potential Energy (Coulomb attraction/repulsion for electrons and nuclei).

A solution to this differential equation is: $\psi(R, t)$

Denominated a wave function, this equation gives the probability of finding any particle (electron or nucleus) in the position $R$ at time $t$. 
Molecular Dynamics

Ideal Calculations (Continued):

The wave equation can be separated in the product of a time independent function and a time dependent propagator:

\[ \psi(R, t) = \psi(R) \times f(t) \]

To find \( \psi(R) \), Schrödinger's equation can be written in a time independent form as:

\[
\left[ \frac{\hbar}{2m} \nabla^2 + V(R) \right] \psi(R) = E \psi(R)
\]

Where \( E \) is the Total Energy, usually expressed via the Hamiltonian Operator \( H \) - the sum of kinetic and Potential \( (V(R)) \) energies, and the del \( (\nabla) \) operator expresses the particle velocity in terms of its position coordinates.

\[
H = -\sum_k \frac{\hbar^2}{2M_k} \nabla^2 - \sum_i \frac{\hbar^2}{2m_i} \nabla^2 + \sum_k \sum_{l>k} \frac{Z_k Z_l e^2}{R_{kl}} + \sum_i \sum_{j>i} \frac{e^2}{r_{ij}} - \sum_{kj} \frac{Z_k e^2}{r_{kj}}
\]
Molecular Dynamics: Approximations?

Why avoid the ideal calculation? Several reasons:

1. **Too hard.** Exact solutions only exist for a reduced number of cases (H, harmonic oscillator, particle in a box, ...). For systems with a large number of electrons, no exact solutions have been found.

2. **Approximate solutions to the Schrödinger equation** can sometimes work well and usually give more “insight” than the exact solutions (e.g. Huckel calculations).

3. **Other approximations** have been tested to work well and make complicated applications accessible for atomistic modeling.

These approximations form the base for what is known today as Molecular Dynamics Simulations.
Molecular Dynamics Simulation: Approximations ...

1. **Born-Oppenheimer. Separates the electronic and the nuclear motions.**
   The systems wave function can then be expressed in terms of an electronic contribution and a nuclear contribution:

   \[ \Psi(R) = \Psi^\text{el}(R) \times \Psi^\text{n}(R) \]

   This is not formally correct, but can be justified based on the fact that the mass of each electron is much less than that of the nucleus. When using a Hamiltonian operator, it can be reduced to:

   \[
   H = -\sum_i \frac{1}{2} \nabla_i^2 - \sum_{kj} \frac{Z_k}{r_{kj}} + \sum_i \sum_{j>i} \frac{1}{r_{ij}}
   \]

   There are some cases, in spite of the difference in mass, for which this approximations cannot be justified. For example, at high energies, or when the energy from two electronic states (two different solutions to Schrödinger's equation) are similar. Ignore these cases for our purpose.
Molecular Dynamics Simulation: Approximations ...

1. Born-Oppenheimer (continued ...).

How does this help?

It introduces the idea of a SURFACE ENERGY POTENTIAL.

For each set of nuclear positions, we can solve Schrödinger's equation to find the energy contributed by the electrons. This energy, and the nuclei-nuclei interactions energy, determine the total potential energy and can be used to find the forces over all atoms.

The electronic energy and the forces can be calculated once for the set of nuclear coordinates and the energies and forces can be extrapolated between these points.

This energy, now only a function of atomic position, is denominated SURFACE ENERGY POTENTIAL.

ELECTRONS HAVE BEEN LEFT OUT OF THE PARTY
Molecular Dynamics Simulation: Approximations ...

2. Treat the nuclei as classical particles moving on the surface energy potential.

This replaces Schrödinger's equation by determining the nuclear positions with classical equations of motion (Newtonian, Lagrangian or other formalism):

\[ F = ma = m \frac{d^2 R}{dt^2} \]

Where the force \( F \) is the surface energy potential force. This is a differential equation that can be solved numerically to determine the trajectories of the nuclei over the surface energy potential.

Consequences: Quantum effects are lost (Tunneling, resonant dispersion, etc.).

\textbf{Ok for elements heavy elements (heavier than He)}
Approximate the surface energy potential using analytical functions to describe the potential energy and the interatomic forces as a function of the spatial coordinates.

This is denominated as a **POTENTIAL ENERGY FUNCTION**.

There are a variety of potential energy functions. Which one to use depends on:

- The type of bond being modeled (metallic, covalent, ionic, among others)
- The desired **precision** (chemical vs. statistical mechanics)
- The desired **transferability** (The ability to describe a large number of bond situations)
- The size and **time** required for the system (10 versus 100 million atoms; 100fs versus 100ns)
- The available computational **resources**

**Most extensive use of molecular dynamics simulations result from the availability of good inter-atomic potential energy functions!**

Schemes: Ad-hoc, Ideas based in QM, Force obtained directly from the quantum mechanical structure calculations
MD: General Procedure

Initialize positions and Velocities
- Temperature/Collision conditions/Structure

Calculate and add forces for each atom

Apply Thermostat / Volume change
- Simple velocity scaling
- White Langevin noise (local heat bath)
- Berendsen friction (global heat bath)
- Hoover/Nose algorithm (formal scaling)

Move atoms forward in time (pico / femto seconds) using Integration Scheme
- Integration Algorithms: Verlet, Gear Predictor-Corrector, Nordsieck Predictor-Corrector

Analyze data
- Animations
- Correlation Functions (statistical mechanics)
- Reaction Probabilities (chemistry)
Thermostats

• **Intended to imitate the interaction between vibrational modes in a system at the macro scale with the dynamic simulations at the atomic scale.**

• While most work by adding or removing heat, the correct selection of a thermostat depends on the system and on the properties of interest.

• Some thermostats compare the total temperature of the system with the desired temperature and adjust the velocity of the particles accordingly. Others couple the motion of each particle to the vibrational modes in a heat bath, without dependencies on the temperature of the system. This last one can remove this vibrational energy trapped in localized modes, an advantageous characteristic in some cases.

• **Hoover:**
  - Adds force terms to maintain a constant kinetic energy (Temperature for large systems).
  - Advantages: Easy to implement, energy (kinetic) conservative.
  - Disadvantages: maintains constant kinetic energy (disassociation of bonds independent of energy).

• **Nosé:**
  - Adds force terms that permit temperature fluctuations.
  - Advantage: Correct modeling of ensemble fluctuations, can remove local undesired or localized correlated motion, couples any number of atoms to a heat bath, reversible dynamics.
Numerical Integrators

- **Basic Idea**: convert a differential equation into a difference equation that can be solved iteratively.

\[ F = ma \quad \text{Newtonian EOM} \]

- **Step 1**: expand \( R(t + \Delta t) \) in a Taylor series:

\[ R(t + \Delta t) = R(t) + \Delta t / 2v(t) + \Delta t^2 / 3a(t) + \ldots \]

- **Step 2**: solve for \( a(t) \), reorganize the series and replace into the Newtonian EOM.

\[ F = ma(t) = m3 / \Delta t^2 \left[ R(t + \Delta t) - R(t) - \Delta t / 2v(t) \right] \]

- **Step 3**: Reorganize and solve for \( R(t + \Delta t) \)

\[ R(t + \Delta t) = R(t) + \Delta t / 2v(t) + \Delta t^2 F / 3m \]

- Given forces, initial velocities and positions, the new position at time \( t + \Delta t \) can be found. In small steps, move the atoms, recalculate the forces and repeat while supervising conservation of energy.
Numerical Integrators

Other algorithms use positions in previous times, instead of velocities, higher order position derivates with respect to time, etc. Those most commonly used are:

**Verlet** (two Taylor expansions, one forward and one backward)

1. \( r_{n+1} = r_n + v_n dt + F_n / 2m \Delta t \) + No velocities; Single F evaluation;
2. \( r_{n-1} = r_n - v_n dt + F_n / 2m \Delta t \) - Large numerical errors (O(\( dt^2 \)) terms is added to an O(\( dt^0 \))
3. \( r_{n+1} = 2r_n - r_{n-1} + F_n / m \Delta t + O(\Delta t^4) \) - Approximated velocities (E calculation) \( v_n = (r_{n+1} - r_{n-1}) / 2dt \)

**Leap Frog** (velocities evaluated at the mid-point of position evaluation and vice versa)

1. \( a(t) dt \) - based on \( r(t) \) + Improved velocity evaluation; Easy control of temp
2. \( v(t+dt/2) = v(t-dt/2) + a(t) dt \) + Reduces error: O(\( dt^1 \)) terms are added to O(\( dt^0 \))
3. \( r(t+dt) = r(t) + v(t+dt/2) dt \) + Instantaneous velocities \( v(t) = (v(t+dt/2) + v(t-dt/2))/2 \)
   - Velocities still approximate

**Velocity Verlet**

1. \( r(t+dt) = r(t) + v(t)dt + a(t)dt^2 \)
2. \( v(t+dt/2) = v(t) + a(t)dt/2 \) mid-point velocity
3. \( a(t+dt) \)
4. \( v(t+dt) = v(t) + [a(t) + a(t+dt)]dt/2 \)

**Predictors-Correctors:** Nordsieck, Gear

**Selection Criteria:** Precision (Chemistry) and stability (statistical mechanics).
Type of Molecular Dynamics Runs

• Characterized by the control exerted on the state functions.
  - **IF:**
  - **GIVES:**
    - \( n_{VE} = \text{constant} \): The **Micro canonical ensemble** of structures. Total energy and volume controlled, the number of atoms constant. Free run, no velocity of pressure scaling. Requires boundary conditions for solvated simulations.
    - \( n_{VT} = \text{constant} \): The **A canonical ensemble**. Average of one atom per step with a temperature bath “solvent molecule”, i.e. assign it a new randomly chose velocity from the correct distribution [Andersen, 1980], or, rescale velocities to maintain temperature.
    - \( n_{PH} = \text{constant} \): The **An isobaric/iso-enthalpic ensemble**. Scale both positions and boundaries by a factor.
Molecular Dynamics Simulation

A 2 step process to define molecular/atomic trajectories:

For a strictly atomistic model

1. Interatomic Forces

\[ F = m_i \cdot \ddot{x}_i = -\nabla_i U(x_1, \ldots, x_n) \]

\[ U = U_r + U_\theta + U_\phi + U_\psi + U_{vdW} + U_{Coulomb} \]

2. Equations of Motion

\[ x_i = x_{i-1} + v_{i-1} \Delta t \]

\[ v_i = v_{i-1} + a_i \Delta t = v_{i-1} - m^{-1} \frac{\partial U(x)}{\partial x} \bigg|_{x_{i-1}} \]

\[ x_2 = x_1 + v_1 \Delta t \]

\[ v_2 = v_1 + m^{-1} F(x_1) \]

for all time steps do

compute intramolecular forces ;
compute non-bonded forces ;
integrate Newton’s equation of motion ;
end do ;

© Andrés Jaramillo-Botero, 2003
# Atomic Interaction Forces: Bonded

<table>
<thead>
<tr>
<th>Description</th>
<th>Illustration</th>
<th>Typical Expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bond Stretch</strong></td>
<td><img src="image" alt="Bond Stretch Illustration" /></td>
<td>$U_r = \frac{1}{2} K_b (R - R_0)^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\frac{\partial U}{\partial R} = K_b (R - R_0)$</td>
</tr>
<tr>
<td><strong>Bond Bend Angle</strong></td>
<td><img src="image" alt="Bond Bend Angle Illustration" /></td>
<td>$U_\theta = \frac{1}{2} C (\cos \theta - \cos \theta_0)^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\frac{\partial U}{\partial \theta} = C (\cos(\theta) - \cos(\theta_0)) \sin(\theta)$</td>
</tr>
<tr>
<td><strong>Dihedral Torsion Angle</strong></td>
<td><img src="image" alt="Dihedral Torsion Angle Illustration" /></td>
<td>$U_\phi = \sum_{n=1}^{p} \frac{1}{2} K_{\theta,n} [1 - d \cos(n\phi)]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\frac{\partial U}{\partial R} = \frac{1}{2} K_\phi \cdot d \cdot n \cdot \sin(n\theta)$</td>
</tr>
<tr>
<td><strong>Inversion</strong></td>
<td><img src="image" alt="Inversion Illustration" /></td>
<td>$U_\psi = \frac{1}{2} C (\cos \psi - \cos \psi_0)^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\frac{\partial U}{\partial \psi} = C (\cos(\psi) - \cos(\psi_0)) \sin(\psi)$</td>
</tr>
</tbody>
</table>

Sub index 0 indicates rest value  
$K_b$: Force Constant (kcal/mol A$^2$)  
$K_{\theta,n}$: Rotational Barrier (kcal/mol), $n$ = periodicity  
$d$: Phase factor
Atomic Interaction Forces: Non-Bond
Distance dependent

Electrostatic Interactions (induce dipoles)

\[ U_{\text{Coulomb}} = C_0 \sum_{i>j} \frac{Q_i Q_j}{\epsilon R_{ij}} \]

\[ U_{\text{electrostatic}} = C_0 \sum_{R_j < R_{\text{cut}}} \frac{Q_i Q_j}{\epsilon R_{ij}} S \delta_{ij}, R_{on}, R_{off} \]

van der Waals

\[ U_{\text{vdW}} = \sum_{R_j < R_{\text{cut}}} U_{\text{vdW}} R_{ij} S R_{ij}, R_{on}, R_{off} \]

Typical Potentials

Lennard-Jones 12-6:

\[ U_{\text{vdW}}(R_{ij}) = AR^{-12} - BR^{-6} = D_0 \left( R_0^{12} - 2 \frac{R_{ij}^{12}}{R_{ij}^{6}} \right) \]

Morse (type 3):

\[ U_{\text{vdW}}(R_{ij}) = D_0 \left( \chi^2 - 2 \chi \right) \]

80-95%

\[ \text{Co} = \text{Conversion Factor (332.0637Kcal/mol)}, \epsilon = \text{Dielectric constant} \]
General Form of the Potential Energy Function vs. Inter-atomic Distance

- Repulsion (nuclei): $1/R$
- Repulsion (internal electrons): $e^{-r}$
- Attraction (valence electrons): $-e^{-r}$
- Dispersion: $1/R^6$
- Nuclear repulsion
- Bonding orbital
- Small overlap

Potential Energy vs. Inter-atomic Distance ($R$)
Atomic Interaction Forces: recapitulation

The potential energy depends on the atomic coordinates and the type of bond in accordance with the nature of the element (type of atom), its hybridization state, and other chemical properties.

\[
F_i = -\nabla \theta_i U = -\left[ \nabla \theta_i U[\theta] + \nabla \theta_i U[x] \right] \\
U = U[\theta] + U[x]
\]

\[
U = \sum_2^{\text{bond stretch}} + \sum_3^{\text{angle bend}} + \sum_{3+}^{\text{stretch} - \text{bend}} \\
+ \sum_3^{\text{stretch} - \text{stretch}} + \sum_{4+}^{\text{bend} - \text{bend}} + \ldots
\]

From the negative gradient of Potential Energy Expression the total Force perceived by each atom is determined ("Force field").
State of the Art in Large Scale MD

With respect to Force Calculations:

- O(N) Multipole Methods (PARALLELIZABLE !!)
  - Cell Multipole Method - CMM
  - Fast Multipole Algorithm

With respect to the Equations of Motion:

- Direct application of Newton's second law of motion and variational principles (Lagrange-Euler)
  (e.g. Mazur y Dorofeed, 1991)

- Constrained high frequency vibrations - SHAKE (van Gunsteren & Berendsen, 1977)

- Rigid Body Dynamics (An Alternate Way to Reduce Computation)
  - NEIMO: Newton-Euler Inverse Mass Operator
  - Floating Base Constrained Force Algorithm
About Multipole Methods

PM D: Mathematical spatial decomposition (voronoi polyhedrals)
CMM: Cartesian decomposition

- Scale all particles into a box
- Divide parent box in half along cartesian axis
- Divide each child to form computational family tree
- Place each particle within a box at lowest level
- Remove empty boxes
- Expand particle charges within lowest level box in multipoles about center of box
- Translate multipole expansion of each lowest level box to center of parent box
- Transform multipole expansions of separated boxes (not nearest neighbours) into Taylor expansions about center of current box
- Translate multipole expansions to Taylor expansion in parent
- Transform Taylor expansion down the tree to child boxes
- Calculate far-field potential via Taylor representation in the particles box
- Calculate interactions between particles that are not well separated at any level
Molecular Dynamics Computation:

- Involves calculating forces on all atoms during each time step, and integrating the forces to update the positions and velocities of the atoms.

- Integration requires a comparatively small fraction of time - completely local operation for atomistic simulations.

- Non-bonded forces decrease as the distance between atoms increases.

- Sufficiently large region of space is likely to be charge-neutral, thus reducing its effect on distant atoms. Use a cutoff radius to save computation time: non-bonded interactions between atoms beyond this distance are not considered. **Short Range.**

- For some simulations, the cumulative effect of far-away interactions cannot be ignored. Even so, a cutoff distance can be used to reduce computation: Forces due to atoms beyond the cutoff distance vary relatively slowly, and therefore can be computed less often, typically only once every 2-5 steps. **Long Range.**

- We focus our attention on cutoff based simulations for decomposition strategies applicable in parallel MD computations.
Parallel MD Computation:

- **Non-bonded computations**: 80-95 percent of the overall computation, depending on the cutoff radius used. (When full-range interactions are included, one can often use a cutoff radius as small as 8Å for the cutoff-only steps, compared with 12 to 18Å used in cutoff-only simulations.)

- **Cannot ignore the computational cost of bonded forces** in designing the decomposition strategy. For example, if one were to completely sequentialize their computations this would affect speedup seriously.

- **NO algorithm is continuously scalable**. Given a simulation of fixed size, there is naturally a limit to the number of processors one can use to achieve higher performance.

- A parallel algorithm is said to be **isoefficiently scalable** if one can increase the number of processors used by it, and still retain its parallel efficiency by increasing the size of the problem being solved.

\[
\text{Efficiency} = \eta = \frac{t_s}{t_p \times p}
\]

- $t_p$: Parallel time
- $t_s$: Serial time
- $p$: Number of Processors
Isoefficiency (Komon & Rao)

- **Efficiency**
  - $s$: problem size, $p$: processors
  - $w(s)$: workload,
  - $h(s,p)$: communication overhead.

  \[
  E_p = \frac{w(s)}{w(s) + h(s,p)}
  \]

- As $p$ grows, communication overhead $h(s,p)$ increases and efficiency $E_p$ decreases.
- For growing $s$, $w(s)$ usually increases much faster than $h(s,n)$.
- *An increase of $w(s)$ may outweigh increase $h(s,p)$ for growing processor number $p$."
- **Question**: For growing $p$, how fast must $s$ grow for efficiency to remain constant?
  - $w(s,p)$ should grow in proportion to $h(s,p)$.
    \[
    E_p = \frac{1}{1 + h(s,p)/w(s,p)}
    \]
- Workload:
  \[
  w(s) = \left[ E_p / (1 - E_p) \right] h(s,p) = Ch(s,p)
  \]
- **Isoefficiency function**: $fE(p) = Ch(s,p)$
- *If workload $w(s)$ grows as fast as $fE(p)$, constant efficiency can be maintained.*
Scalability

• Typically when $p$ is increased, $\eta$ will normally decrease (due to increase in parallelization overheads). But often when we increase $N$, the ratio of communication ($C$) to computation ($R$) decreases, leading to improved $\eta$.

• For an algorithm to be scalable $\eta_1 \leq \eta_2$ must hold true for any two different problem sizes, $N_1 < N_2$, running on $p_1 < p_2$.

$$ t_p = \frac{t_s}{p} + t_{\text{overhead}} = \frac{t_s}{p} \left( 1 + \gamma \right) $$

• Where
  - $\gamma$: communication to computation ratio

$$ \frac{t_{s1}}{t_{p1} \times p_1} = \frac{t_{s2}}{t_{p2} \times p_2} \rightarrow \frac{t_{s1}}{t_{p1} \times p_1} = \frac{t_{s2}}{t_{p2} \times p_2} \rightarrow \gamma_1 = \gamma_2 $$

• Thus, for $\eta$ to remain the same while increasing $p$, we must be able to find a larger $N$ for which $\gamma$ remains the same.
Speedup: Amdahl’s Law

P: Parallel Fraction
S: Serial Fraction
N: Number of Processors

\[ \text{Speedup} = \frac{1}{1 - P} \quad \text{Speedup} = \frac{1}{S - P / N} \]

\( f \) is parallel fraction
\( S \) is serial fraction
\( N \) is number of processors

© Andrés Jaramillo-Botero, 2003
MD Parallelization: General Methods

1. Data Replication \cong Atom Decomposition:

Atom decomposition involves a subgroup of atoms being assigned to each processor, the processor computes forces on its atoms no matter where they move in the simulation domain, hence the name “atom decomposition”. \(O(N/p)\) Computation – Communication Cost \(O(N)\), independent of \(p\)


3. Force Decomposition: Block splitting of matrix representing all combinations of atom pairs. Avoid total exchange but, do not take into account locality of data for calculating intermolecular forces. Not scalable. Computation Cost: \(O(N/P)\) – Communication Cost: \(O(N/sqrt(P))\)


4. Spatial Decomposition: Geometric decomposition of domain. Assign a part of the domain to a processor. Atoms allowed to move between neighboring processors. Computation Cost: \(O(N/P)\) – Communication Cost: \(O(N/P)\)

Data Replication

- Replicate all data (arrays containing attributes, coordinates and forces of atoms) on each processor.

- Force computations can be evenly distributed across processors, as any processor is capable of carrying out any particular force computation.

- If there are $N$ atoms, and $p$ processors, the $O(N)$ forces accumulated by each processor must be added up across all processors. This requires communication time proportional to $N \log p$. Assuming cutoff, the amount of computation required is proportional to $N$ (with a large proportionality constant), leading to $N/p$ computation per processor.

- As the parallel execution time of the program is the sum of its computation and communication times (minus overlap), the parallel efficiency is affected by the communication time. However, $\gamma$ for replicated data is $p \log p$, and is independent of $N$. So, if we want to simulate a system twice as large as the current one, we cannot hope to double the number of processors to retain the same efficiency, because the fraction of time spent in communication will be larger. Thus, this decomposition is said to be non-scalable.

- In practice, it works effectively for tens of processors, and has the further advantage of being easier to implement, especially for an existing serial program. As a result, this strategy is used by many production quality MD programs.
Atom Decomposition (AD)

1. Each processor has a complete copy of the coordinates and velocities of all the atoms in the system.

2. Each processor is assigned a sub-block of the N*N force matrix (N is the number of atoms) to calculate. Each processor calculates N(N-1)/2P of these interactions (Newton’s 3rd law).

3. No processor has complete force matrices (each requires contributions from others). The incomplete force arrays must be circulated to all the other processors to complete the summation of the forces on each processor. This requires a “global pass-and-sum” (Smith, 1991).

4. Integrate EOM independently on each processor.
Force Decomposition (FD)

Two Types:

1. **Force matrix.** Distributes the sparse force matrix in a block-wise fashion across processors (as opposed to row-wiser decomposition).
   - Each processor's share of the NxN force matrix is a block of size \((N/vp)x(N/vp)\). To compute this block the processor needs the coordinates of \(2N/vp\) atoms, which come from \(vp\) different processors.
   - The communication cost per processor is thus \(O(N/vp)\) leading to the communication to computation ratio of \(vp\).
   - Much better than RD, but still not scalable: as we increase the number of processors, the proportion of communication cost increases.
   - Plimpton and Hwang *et al* shows that this method provides a better speedup than RD, (good speedups up to hundreds of processors). Results obtained by Hwang show that the proportion of communication cost goes from 6% : 32 processors to 36%: 128 processors.

2. **Systolic loop algorithm** *(Raine et al., 1989)*
   - Systolic Loop Double Group (SLD-G)
   - Systolic Loop Bidirectional Group (SLB-G)
   - Systolic Loop Single Group (SLS-G)
Force Decomposition (FD)

- As the force matrix is sparse, and the atoms that are next to each other in the array are typically physically closer (the converse is false), one may get a non-uniform distribution of work with FD.

- Plimpton et al. -> random reordering of atoms to eliminate spatial locality, and restore load balance. (To handle symmetry arising out of Newton's third law, they compute the interaction between atoms \( i \) and \( j \) in one of the two possible blocks containing \( F[i,j] \) and \( F[j,i] \) respectively, depending on whether \( i+j \) is even.)

- Hwang et al. use recursive bisection to partition the atoms so that nearby atoms are assigned to the same processor, and reorder the force matrix accordingly. The resultant load imbalance is handled by assigning an irregular-shaped piece of force matrix to each processor. Empirically, the communication costs are seen to be smaller than AD.
FD: Systolic Loop Double Group

The general concept involves “packets” of data being circulated between processors, with the packets containing data relating to a subset of atoms (e.g. the atom coordinates, velocities and force accumulators). **Ring Topology** with odd $p$.

Each $p$

1. Each processor duplicates its packet (coordinate arrays and force accumulators=0). One remains fixed, the other is passed in a single direction.
2. The pair forces within a home packet are calculated and added to the home force accumulators.
3. The duplicated packet is then passed to the next processor, hence each processor $\rightarrow$ atomic coordinates and force accumulators for two packets. Calculate forces between groups and add to force accumulators.
4. Pass packets again in the same direction to the next processor- process completed.
5. Pass duplicated packets back to home processor and add replicated and home force packets.

$p \rightarrow (p-1)/2$ pass times so all possible pair forces have been calculated (reason for an odd $p$).
FD: Systolic Loop Bidirectional Group

Same as before, but each of the packets on a processor are sent in opposite directions i.e. there is no “home” packet. At the end of \((p-1)/2\) data passes the duplicate data packets are within one pass of each other and hence the “rewind” step is much shorter than for the SLD-G algorithm.
FD: Systolic Loop Single Group

Each of the two packets on a node represent different groups of atoms. The number of processors can be odd or even and the processors are connected in a line with a “head” and “tail” processor at either end.

More generally applicable than the previous two methods.

5. Tail processor sends 1\textsuperscript{st} data package to the left, this is then replaced by the 2\textsuperscript{nd} data package on the same processor. The tail processor receives one package from left.

1. Forces within packets are calculated and then the forces between different packets on processor calculated.

2. Exchange data packets.

3. Each processor (other than head or tail) sends 1\textsuperscript{st} data package to the right and receives one from the left.

4. The processor then sends the second package to the left and receives one from the right.
Spatial Decomposition
Assigning nearby atoms to the same processor

There are broadly three ways of doing this:

1. Partitioning space into $p$ boxes, one per processor.

With sufficiently large $N$, communication cost is proportional to the surface of the box, while the computation proportional to its volume (assuming constant density). Highly scalable. Hard to use if $p$ cannot be factored into three roughly equal factors.
Spatial Decomposition

2. Partitioning space into fixed-size boxes, with dimension larger than the cutoff distance, requiring communication only between neighboring boxes.

- Each box needs to communicate with a constant number of neighboring boxes.
- Communication is proportional to $N/p$, and the algorithm is scalable with linear isoefficiency.
- Load imbalance can be a severe problem for SD, especially for simulating non-periodic systems.
Spatial Decomposition

3. Partitioning space into a large number of small boxes, requiring each box to communicate with a large number of boxes.

- Clark et al

- involves using boxes with sizes smaller than the cutoff distance, so that each box needs data from non-neighboring boxes. Multiple boxes can be mapped to each processor.

- The large number of messages resulting from the scheme can be reduced (but not the volume of data transferred).

a) east/west exchanges

b) north/south exchanges

c) up/down exchanges
Spatial Decomposition

• Features:
  – Short range potential cut-off ($r_{\text{cut}} \ll L_{\text{cell}}$)
  – Spatial decomposition of atoms into domains
  – Map domains into processors
  – Calculate forces, solve EOM
  – Re-allocate atoms leaving domains

• Advantages:
  – Good load balancing
  – Ideal for huge systems
  – Simple communication structure
  – Fully distributed memory requirement
  – Dynamic load balancing possible

• Disadvantages:
  – Problems with mapping/portability
  – Requires short potential cut-off
  – Complex force fields tricky
Scalability of decomposition strategies with examples of programs that use them.

<table>
<thead>
<tr>
<th>Method</th>
<th>Communication costs per node</th>
<th>Theoretical Scalability</th>
<th>Comment</th>
<th>Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD</td>
<td>$O(N \log p)$</td>
<td>No</td>
<td>Easy</td>
<td>CHARMM, AMBER, UHGromos, X-Plor</td>
</tr>
<tr>
<td>AD</td>
<td>$O(N)$</td>
<td>No</td>
<td></td>
<td>EGO</td>
</tr>
<tr>
<td>FD</td>
<td>$O(N / \sqrt{p})$</td>
<td>No</td>
<td>Tolerable Communication</td>
<td>LAMMPS, CHARMM</td>
</tr>
<tr>
<td>SD</td>
<td>$(N/p)^{2/3}$</td>
<td>Isoefficient</td>
<td>Load imbalance</td>
<td>Ddgmq, SIGMA</td>
</tr>
<tr>
<td>QSD</td>
<td>$O(N/p)$</td>
<td>Isoefficient</td>
<td>Load imbalance</td>
<td>NAMD, PMD, CMM, EulerGromos</td>
</tr>
<tr>
<td>QSD+FD</td>
<td>$O(N/P)$</td>
<td>Isoefficient</td>
<td>Load balance feasible</td>
<td>NAMD2</td>
</tr>
</tbody>
</table>

Parallel Computational Complexities

<table>
<thead>
<tr>
<th>Method</th>
<th>Arithmetic cost</th>
<th>Communication cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atom decomposition</td>
<td>$O(N/p)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>Force decomposition</td>
<td>$O(N/p)$</td>
<td>$O(N/\sqrt{p})$</td>
</tr>
<tr>
<td>Spatial decomposition</td>
<td>$O(N/p)$</td>
<td>$O(N/p)$</td>
</tr>
</tbody>
</table>
Other Parallelization Schemes


---

Figure 4  Schematic illustration of the parallel replica method. The four steps, described in the text are (a) replication of the system into $M$ copies, (b) dephasing of the replicas, (c) independent trajectories until a transition is detected in any of the replicas, and (d) brief continuation of the transitioning trajectory to allow for correlated events such as recrossings or follow-on transitions to other states. The resulting configuration is the replicated, beginning the process again.

Figure 5  Nine snapshots from a parallel replica simulation of an island on top of an island on the Ag(111) surface at $T=480$ K. On a microsecond time scale, the upper island gives up all its atoms to the lower island, filling vacancies and kinks, or it does so. This simulation took five days to reach one microsecond on a 32-GHz Pentium III processor.
**Parallel Constrained Rigid Body Dynamics**

**Motivation**

- **Polymers and proteins**: conformational analysis (introduction of constraints)
- Drop in the number of DOF from $3N$ to $N$
- Polyethylene: $C_pH_{2p+2}$ $\Rightarrow$ $3N = 9p + 6$ $\Rightarrow$ torsional DOF is $N = p-1$
- Simplified analysis
- Increased Time Steps (orders of magnitude speedups)
- Constrained Dynamics (EOM)
- Serial: Linearly scales with $N$ $\Rightarrow$ E.g. 1001 atoms $\Rightarrow$ 333
- Parallel: Log scale with $N$ $\Rightarrow$ 1024 atoms $\Rightarrow$ 10
Why Parallelization of EOM is Required?

Amdahl ...
5% serial can kill parallel efficiency and scalability

**REQUIREMENT:**
Significant Improvements in computational efficiency (orders of magnitude !!!).

- Strictly parallel algorithms for systems with a serial chain topology, resulting in a time, space and processor optimal solution to the Equations of Motion (inverse and forward problems). These are $O(\log_2 N)$ algorithms when using $O(N)$ processors.
- Highly efficient for practical implementation on massively parallel MIMD architectures due to its coarse grain size and simple communication structure.
- Application to large-scale, long-term rigid-body MD simulations of systems with serial chain topology, e.g. Polyethylene (PE). With $P < N$ processors. Optimally implemented in $O(N/P + \log_2 P)$ with $O(P)$.
- Efficient variants for systems with complex topologies, such as branched, grafted polymers and hyper-branched dendrimers. Achieves optimal computation time of $O(\log_2 N + P)$ when using $O(N)$ processors.
Rigid Body Molecular Dynamics

\[
f_{C_i} = \sum_{i=1}^{n} \left( x_o - x_i \right) \times f_i
\]

\[
M_{\text{cuerpo}} = \sum_{i=1}^{\text{átomos}} m_i
\]

- No intra-body bond or non-bond interactions.
- Only inter-body interactions are considered.

© Andrés Jaramillo-Botero, 2003
## Equations of Motion (n-body system)

### 1. Interatomic Forces

#### Atomistic (Cartesian) Model

\[ M_c \ddot{X} = F_c \]

- **Characteristics**
  - Cartesian Coordinates
  - \( M \) diagonal and scalar
  - No motion constraints (ODE)

- **Consequences**
  - Short integration time steps
  - Difficult to deal with constraints
  - Small scale

#### Internal Coordinates Model

\[ M(Q)\ddot{Q} = F_T(Q, \dot{Q}) \]

- **Characteristics**
  - Internal Coordinates
  - \( M \) dense
  - Constrained System (CADE)

- **Consequences**
  - Longer integration time steps
  - Implicit constraints
  - Increment on scale

---

**Challenge**

*FASTER AND MORE EFFICIENT ALGORITHMS*

---

Cluster Force Fields

Brandt et al, 2000

---

© Andrés Jaramillo-Bozeto, 2003
Equations of Motion for Serially Coupled Rigid Bodies

\[ F_T \theta, \dot{\theta} = \Gamma \theta ( -\xi \theta, \dot{\theta} ) = M \theta \ddot{\theta} \]

\[ \ddot{\theta} = M^{-1} [ \Gamma - \xi ] \]

- **Effective Force vector** \( F_T \)
- **Generalized force vector** (derived from the potential energy function) \( \Gamma \)
- **Velocity dependent forces** (Coriolis, Centrifugal, Gyroscopic) \( \xi \)
- **Effective state** (effective position, velocity, and acceleration vectors) \( \theta, \dot{\theta}, \ddot{\theta} \)
- **Inverse Articulated Body Inertia Matrix** \( M^{-1} \)

**Conventional Algorithms for the Inversion of the Mass Matrix:**

- **O(N^3):** Explicit calculation of \( M \) in \( O(N^2) \) and of \( M^{-1} \) in \( O(N^3) \)
- **O(N^2):** Indirect/Direct Methods for computing \( M \) and \( M^{-1} \)
- **O(N):** Factorization and recursive inversion of \( M \)
  - Optimum for serial implementations
  - Strictly Serial (NOT scalable under parallelism)
Articulated Body Inertia: Inability of existing O(n) algorithms to scale with parallelism

**PROBLEM:** The resulting expression for the inverse mass operator, in the existing O(n) algorithms for the solution to the Equations of Motion of coupled multibodies, contain non-linear recurrences of order > 1 (including NEIMO).

\[ I_i^{AR} = I_i + I_{i+1}^{AR} - \left[ \varphi_1(I_{i+1}^{AR}) \right]^{-1} \varphi_2(I_{i+1}^{AR}) = I_i + I_{i+1}^{AR} - \phi(I_{i+1}^{AR}) \quad i = n...1 \]

Where, \( D_i \) is constant, \( \varphi_1 \) y \( \varphi_2 \) are polynomials of 1\(^{st}\) and 2\(^{nd}\) order, and \( \deg \phi = \max(\deg \varphi_1, \deg \varphi_2) = 2 \)

**Bounded** parallelism in these type of recurrences with \( \deg \phi > 1 \)

**No algebraic transformation exists** for parallel solutions of this type.

**REQUIRED**

An optimum time and space parallel algorithm for solving the equations of motion: for the physical problem this implies that, for an n body system, the computational complexity must be \( O(\log_2 n) \).

Under strict parallelism this must then follow for \( O(n) \) processors.

**Such algorithm can only be derived from a serial O(N) algorithm!!**
Recapping on the Problem at Hand

Conventional algorithms for MD simulation still limited with respect to:

- Time scales,
- Size,
- Scalability,
- Computational Complexity,
- Complex Topological Systems.

Proposed Solution

Design, develop and prove a new recursive formulation for the equations of motion for large scale, long-term Molecular Dynamics Simulations of Complex Systems taking into account the following criteria:

- Strictly parallel and highly scalable,
- Time and space efficient (and processor), and
- Of minimum computation complexity.
Rigid Body Dynamics: Spatial Notation

Spatial Expressions

<table>
<thead>
<tr>
<th>Physical Quantities $\in \mathbb{R}^{6\times1}$ and $\in \mathbb{R}^{6\times6}$</th>
<th>Transformation Operators $\in \mathbb{R}^{6\times6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V = \mathbf{W}_V$</td>
<td>$\mathbf{R}_{i,i+1} = \mathbf{M}<em>i^{-1} 0 \mathbf{P}</em>{i,i+1}^\mathbf{p}$</td>
</tr>
<tr>
<td>$\dot{V} = \mathbf{W}_V$</td>
<td>$\mathbf{F}<em>{i} = \mathbf{S}</em>{O_i,cm}^T \mathbf{N}_{O_i}$</td>
</tr>
<tr>
<td>$F_{i} = \mathbf{S}<em>{O_i,cm}^T \mathbf{F}</em>{i,cm} = \mathbf{I}<em>{O_i} \dot{V}</em>{O_i} - \dot{\mathbf{S}}<em>{O_i,cm} \mathbf{I}</em>{O_i} \mathbf{V}<em>{O_i} + \mathbf{I}</em>{O_i} \dot{\mathbf{V}}_{O_i}$</td>
<td></td>
</tr>
</tbody>
</table>

Single Body Dynamics using Spatial Operator Algebra

Conventional Representation

Velocity

$\mathbf{W}_{i,cm} = \mathbf{W}_0 \mathbf{O}_i$

$\mathbf{V}_{i,cm} = \mathbf{V}_0 \mathbf{O}_i + \mathbf{W}_0 \mathbf{O}_i \times \mathbf{S}_{O_i,cm}$

$\mathbf{W}_{i,cm} = \mathbf{W}_0 \mathbf{O}_i$

$\mathbf{V}_{i,cm} = \dot{\mathbf{V}}_0 \mathbf{O}_i + \mathbf{W}_0 \mathbf{O}_i \times \mathbf{S}_{O_i,cm} +$

$\mathbf{W}_0 \mathbf{O}_i \times \mathbf{w}_0 \mathbf{O}_i \times \mathbf{S}_{O_i,cm}$

$\mathbf{V}_{i,cm} = \dot{\mathbf{V}}_0 \mathbf{O}_i + \mathbf{W}_0 \mathbf{O}_i \times \mathbf{S}_{O_i,cm} +$

$\mathbf{W}_0 \mathbf{O}_i \times \mathbf{w}_0 \mathbf{O}_i \times \mathbf{S}_{O_i,cm}$

Spatial Representation

Acceleration

$\mathbf{N}_{O_i} = \mathbf{N}_{O_i,cm} + \mathbf{S}_{O_i,cm} \times \mathbf{f}_{i,cm} = \mathbf{J}_{O_i} \dot{\mathbf{w}}_0 \mathbf{O}_i + \mathbf{m}_i \mathbf{S}_{O_i,cm} \times \mathbf{v}_i + \mathbf{w}_0 \mathbf{O}_i \times \mathbf{J}_{O_i} \dot{\mathbf{w}}_0 \mathbf{O}_i$

$\mathbf{f}_{O_i} = \mathbf{f}_{cm} = -\mathbf{m}_i \mathbf{S}_{O_i,cm} \times \dot{\mathbf{w}}_0 \mathbf{O}_i + \mathbf{m}_i \mathbf{v}_i \mathbf{O}_i + \mathbf{m}_i \mathbf{w}_0 \mathbf{O}_i \times (\mathbf{w}_0 \mathbf{O}_i \times \mathbf{S}_{O_i,cm})$

$\mathbf{F}_{O_i} = \mathbf{S}_{O_i,cm} \mathbf{F}_{i,cm} = \mathbf{I}_{O_i} \dot{\mathbf{V}}_0 \mathbf{O}_i - \dot{\mathbf{S}}_{O_i,cm} \mathbf{I}_{O_i} \mathbf{V}_0 \mathbf{O}_i + \mathbf{I}_{O_i} \dot{\mathbf{V}}_0 \mathbf{O}_i$
Equations of Motion for Serially Articulated Rigid Multibody Systems

**INVERSE DYNAMICS**: Recursive serial O(N) Newton-Euler Algorithm for N Body Serial Chains.

1. **Spatial Velocities** for body \( i \) (\( i=1...n \))

\[
V_i = \dot{P}_{i-1,i}^T V_{i-1} + H_i \dot{Q}_i
\]

2. **Spatial Accelerations** for body \( i \) (\( i=1...n \))

\[
\dot{V}_i = \dot{P}_{i-1,i}^T \dot{V}_{i-1} + H_i \ddot{Q}_i + \dot{P}_{i-1,i}^T V_{i-1} + \dot{H}_i \dot{Q}_i
\]

3. Using d’Alambert principle, define **Spatial Forces** acting on body \( i \) (\( i=n...1 \))

\[
F_i = \dot{P}_{i,i+1} F_{i+1} + F_{i,cm}
\]

\[
F_i = \dot{P}_{i,i+1} F_{i+1} + I_i \dot{V}_i + \dot{I}_i V_i + I_i \dot{S}_{O_i,cm}^T V_i
\]

\[
F_i = \dot{P}_{i,i+1} F_{i+1} + I_i \dot{V}_i + \boldsymbol{\phi}_i
\]

The effective forces are then a projection onto the DOF (special case, floating base, \( H=U \))

\[
F_{Ti} = H_i^T F_i
\]

Neglecting: Gravitational forces and tip contact forces. These can be included locally to every \( i \)
Finding Effective Accelerations $\ddot{Q}$ from $F_T$

**FORWARD DYNAMICS: CONSTRAINT FORCE ALGORITHM FOR MD SIMULATIONS**

$O(N) \longrightarrow O(\log_2 N)$ (Fijany, Jaramillo-Botero, Cagin, Goddard, APS, March 1997).

**Fundamental Motivation**

Force decomposition scheme that results in the direct factorization of $M^{-1}$ in a Schur Complement form, which will then be demonstrated to scale under strict parallel computations in minimum order complexity.

1. From the Linearized (no-velocity) Newton-Euler Equations of Motion

   \[ P^T \dot{V} = H \cdot \dot{Q} \quad PF = I \dot{V} \]

2. Decompose total forces into orthogonal spaces:

   \[ F = HF_T + WF_S \]
   \[ HH^T + WW^T = U \quad H^T H = U \quad W^T W = U \quad W^T H = 0 \quad H^T W = 0 \]

3. Express EOM in terms of Effective and Constraint Forces:

   \[ BF_T + AF_S = 0 \quad y \quad CF_T + B^T F_S = \ddot{Q} \]

4. Solve for Effective Accelerations

   \[ \ddot{Q} = [C-B^T A^{-1} B] F_T \Rightarrow M^{-1} = [C-B^T A^{-1} B] \]

**Direct Schur Complement form factorization of $M^{-1}$**

For serial chains, $A$, $B$, and $C$ are block Tridiagonal matrices ($A,C$ are SPD)
Inverse Dynamics: Strictly Parallel in $O(\log_2 N)$

Expressing Newton-Euler EOM for Serial Chains as a 1st Order Linear Inhomogeneous Recurrence

1. **Spatial Velocities** for body $i$
   
   Initial Conditions ($n=P=i$) $j=\lceil \log_2 n \rceil$
   
   \[
   V_i^0 = H_i \dot{q}_i \quad P_i^{0T} = P_i^T R_i^T
   \]

   \[
   V_i^j = V_i^{j-1} + P_i^{j-1T} V_i^{j-1}
   \]

   \[
   P_i^{jT} = P_i^{j-1T} P_i^{j-1T}
   \]

   Equivalent to $P_i^{jT} = P_{i+2^{j-1}} P_i^{j-1T}$

2. **Spatial Accelerations** for body $i$
   
   Same structure as for Spatial Velocities but no need to send/receive $P$, terms. Initial Condition:
   
   \[
   \ddot{V}_i^0 = H_i \ddot{q}_i + \dot{H}_i \dot{q}_i + \dot{P}_i^{T} V_i^{\lceil \log_2 n \rceil}
   \]

3. **Spatial Forces** acting on body $i$
   
   Initial Conditions, assuming $F_c$ is local to every $i$
   
   \[
   P_i^0 = \varpi_i^{T} \mathbf{j} \quad F_i^0 = F_i + I_i \dot{V}_i^{\lceil \log_2 n \rceil} + \varpi_i
   \]

   \[
   F_i^j = F_i^{j-1} + P_i^{j-1T} F_i^{j-1}
   \]

   \[
   P_i^j = \varpi_i^{T} \varpi_i^{j-1T} P_i^{j-1T}
   \]

   \[
   F_{Ti} = H_i^{T} F_i^{1}
   \]

**NOTE:** Spatial Accelerations can be computed concurrently with Spatial Velocities

$O(N/P + \log_2 P) \quad N \geq P$
Forward Dynamics for MD Simulations

Free-Floating Base case: extended constraint force algorithm

\[ \ddot{Q} = CF_T - B^T [ A^{-1} (BF_T - \dot{R}_b) ] - \ddot{Q}_b \]

\[ BF_T + A \dot{F}_S = \dot{R}_b \]
\[ CF_T + B^T F_S = \ddot{Q} + \ddot{Q}_b \]

\[ A' = A + \text{diag} [v_b] \]

\[ \dot{V}_b = [0 \ 0 \ \ldots \ 0 \ W^T \hat{p}_o^T I_o^{-1} \hat{p}_o W]^T \]
\[ \dot{R}_b = [0 \ 0 \ \ldots \ 0 \ W^T \hat{p}_o^T I_o^{-1} \left( F_o - \hat{p}_o H_F F_R \right)]^T \]

### Computational Step

**O** Compute \( F_T = \Gamma - \xi \)

**Parallel Computation Complexity**

\( O(\log_2 n) \) in \( O(n) \) Processors

Begin DO PARALLEL

1. \( \ddot{Q}^1 = CF_T ; \quad S = BF_T - \dot{R}_b \)

2. \( F_s = - A'^{-1} S \)

3. \( \ddot{Q}^2 = B^T F_S ; \quad \ddot{Q}_b \)

4. \( \ddot{Q} = \ddot{Q}^1 + \ddot{Q}^2 - \ddot{Q}_b \)

End

\( O(\log_2 n) \) in \( O(n) \) Processors

\( O(1) \) in \( O(n) \) Processors
Finding Fs in $O(\log_2 N)$: Modified Cyclic Reduction Scheme

Thru data redistribution in a strictly parallel implementation, an homogeneous grain size and a full 100% (theoretical) overlap between run-time costs and communication costs is achieved for $n=P$ cases. For $n>P$ a conventional CR scheme is used.

Reduction of each row equation, $[i-2^{-1}, i, i+2^{-1}]$ for $j = 1 \ldots \left\lfloor \log_2 n \right\rfloor$

\[
\begin{align*}
L_{i-2} x_{i-2} + D_{i-1} x_{i-1} + U_{i-1} x_i &= b_{i-1} \\
L_{i-1} x_{i-1} + D_i x_i + U_i x_{i+1} &= b_i \\
L_i x_i + D_{i+1} x_{i+1} + U_{i+1} x_{i+2} &= b_{i+1}
\end{align*}
\]

For symmetric tridiagonal systems: $L = U^T$

DO PARALLEL

\[
\begin{align*}
D_i^j = D_i^{j-1} - U_i^{j-1} (D_{i+2}^{j-1})^{-1} U_i^{j-1} &+ U_i^{j-1} T (D_{i+1}^{j-1})^{-1} U_i^{j-1} \\
b_i^j = b_i^{j-1} - U_i^{j-1} (D_{i+2}^{j-1})^{-1} b_i^{j-1} &+ U_i^{j-1} T (D_{i+1}^{j-1})^{-1} b_i^{j-1} \\
U_i^j = -U_i^{j-1} (D_{i+2}^{j-1})^{-1} U_i^{j-1} &+ U_i^{j-1} T (D_{i+1}^{j-1})^{-1} U_i^{j-1}
\end{align*}
\]

After $r = \left\lfloor \log_2 n \right\rfloor$ steps, the tridiagonal system of equations is fully diagonalized hence the solution for $x \ (F_s)$ becomes trivial: $x_i = D_i^{(r)-1} b_i^{(r)}$

Support for: Molecular Hyper-branched Structures

1. For an \( n \) body system and \( b \) branches, define new spatial operators (velocity, acceleration, force, effective accelerations, inertia, projection ...) as follows:

\[
\begin{align*}
V_1 & \quad \dot{V}_1 & \quad F_1 \\
V_2 & \quad \dot{V}_2 & \quad F_2 \\
\vdots & \quad \vdots & \quad \vdots \\
V_b & \quad \dot{V}_b & \quad F_b \\
\end{align*}
\]

\[
\ddot{\mathbf{Q}}' = \begin{bmatrix} \ddot{Q}_1 & \ddot{Q}_2 & \ldots & \ddot{Q}_b \end{bmatrix}^T
\]

\[
\mathbf{Y}' = \text{diag } I_1 \quad I_2 \quad \ldots \quad I_b
\]

\[
\mathbf{H}' = \text{diag } H_1 \quad H_2 \quad \ldots \quad H_b
\]

2. Define interconnection matrices (for branches and bodies)

\[
\mathbf{P} = \begin{bmatrix}
P_{p,1} & 0 & \ldots & 0 & \hat{P}_1 \\
0 & P_{p,2} & 0 & \ldots & \hat{P}_2 \\
\rho & 0 & \ldots & 0 & \hat{P}_3 \\
P_{4,1} & P_{4,2} & 0 & \ldots & \hat{P}_4 \\
0 & 0 & P_{5,3} & P_{5,4} & \hat{P}_5 \\
\end{bmatrix}
\]

3. Solve the new equations of motion given by:

\[
\mathbf{H}'\ddot{\mathbf{Q}}' = \rho^T \ddot{\mathbf{V}}' \\
\rho \mathbf{F}' = \mathbf{I}' \ddot{\mathbf{V}}'
\]
Support for: Closed Chain Molecular Structures

1. Splitting the closed chain into 2 sub serial chains \((n1 \text{ y } n2)\), and applying the following initial base conditions:

\[
f_{1,n1} = -f_{2,n2} = f
\]
\[
\dot{V}_{1,n1} = \dot{V}_{2,n2}
\]

2. The equations of motion for each sub serial chain are then given by \((i=1,2)\):

\[
M_i \ddot{Q}_i + \xi_i = \Gamma_i - J_i^T f
\]

3. Then, the spatial acceleration for each sub serial chain can be found to be given by:

\[
\ddot{Q}_i = M_i^{-1} (\Gamma_i - \xi_i) + M_i^{-1} [J_i^T f]
\]
\[
= \ddot{Q}_{if} + C \ddot{Q}_i
\]
\[
\dot{Q}_{if} = M_i^{-1} (\Gamma_i - \xi_i) \quad \text{Solved!}
\]
\[
\Delta \ddot{Q}_i = M_i^{-1} J_i^T f \quad \text{Continue to step 4}
\]

4. Differentiating the spatial velocity for each chain, then replacing result from step 3 and using the operational space mass matrix [Khatib] to find the tip force, \(f\), then from step 3 the joint acceleration correction factor (inter-chain net spatial accelerations) follows

\[
\Lambda_i^{-1} = J_i M_i^{-1} J_i^T
\]
\[
\dot{V}_{i,ni} = J_i \ddot{Q}_i + J_i \dot{Q}_i
\]
\[
\dot{V}_{i,ni} = J_i [\ddot{Q}_{if} + C \ddot{Q}_i] + J_i \Delta \ddot{Q}_i
\]
\[
\dot{V}_{if} = J_i \ddot{Q}_{if} + J_i \Delta \ddot{Q}_i
\]
\[
\dot{V}_{i,ni} = \dot{V}_{if} + J_i \Delta \ddot{Q}_i
\]
\[
\dot{V}_{i,ni} = \dot{V}_{if} + \Lambda_i^{-1} f
\]

\[
f = (\Lambda_2^{-1} - \Lambda_1^{-1} [\dot{V}_{2f} - \dot{V}_{1f}])
\]

Results: Hardware and Software Setups

MPPs
- Cray T3E-300 (256): San Diego Supercomputing Center
- Cray T3D (256): Jet Propulsion Laboratory
- Intel / Paragon (512): California Institute of Technology
- SGI Origin 2000 (16): Materials and Process Simulation Center (MSC-Caltech)
- IBM SP2 (128): University of Texas, Austin

Other Architectures Employed
- Dual PPro200 SMP with Linux Redhat 4.1 (Personal Development Platform)

Programming Paradigm - SPMD
- nx (Intel), native MPI, and MPICH (no parallel I/O supported)

Debugging and Profiling
Paragraph (nx), pdb (nx), Upshot (MPICH), Totalview (Cray-SGI), gdb (Linux), “printf”
Results: Computation Times per Stage

Pre-processing
NE [1]: Spatial Velocities
NE [2]: Spatial Accelerations
NE [3]: Spatial and Effective Forces
CFA [1]: Inter-body Active Force Component
CFA [2]: Inter-body Constraint Forces
CFA [3]: Effective Accelerations

n = P = 16

n = P = 128

Cray T3E-300
Results: Speedups

ECFA (N=120) Complete

Speedup for N=P (Amdahl)

ECFA (N=16) Without Cyclic Reduction

Speedup for N=P (Amdahl)
Results: Communication Structure

NE: n = P = 4

NE + ECFA: n = P = 4
Results: Grain Size (R/C Ratio)

R/C Factor for varying P and N/P ratio
(without data packing)

<table>
<thead>
<tr>
<th>n/P</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.49227</td>
<td>1.291825</td>
<td>1.493204</td>
<td>1.634975</td>
<td>1.298147</td>
</tr>
<tr>
<td>2</td>
<td>1.77596</td>
<td>1.808633</td>
<td>1.743118</td>
<td>1.586873</td>
<td>1.296485</td>
</tr>
<tr>
<td>4</td>
<td>2.17205</td>
<td>1.914455</td>
<td>1.846568</td>
<td>1.658595</td>
<td>1.81669</td>
</tr>
<tr>
<td>8</td>
<td>2.45631</td>
<td>2.752522</td>
<td>2.164283</td>
<td>1.952869</td>
<td>1.417508</td>
</tr>
<tr>
<td>16</td>
<td>3.44145</td>
<td>2.699049</td>
<td>2.036946</td>
<td>1.934904</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>3.48878</td>
<td>3.055212</td>
<td>2.523531</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>3.50945</td>
<td>3.064716</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

R/C Factor for varying P and N/P ratio
(with data packing)

<table>
<thead>
<tr>
<th>n/P</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.88114</td>
<td>1.9434</td>
<td>1.866876</td>
<td>1.658365</td>
<td>1.31597</td>
</tr>
<tr>
<td>4</td>
<td>2.42531</td>
<td>2.088669</td>
<td>1.993677</td>
<td>1.759399</td>
<td>1.359799</td>
</tr>
<tr>
<td>8</td>
<td>2.84539</td>
<td>3.364737</td>
<td>2.467136</td>
<td>2.171037</td>
<td>1.459251</td>
</tr>
<tr>
<td>16</td>
<td>4.37268</td>
<td>3.254591</td>
<td>2.154693</td>
<td>2.138228</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>4.41878</td>
<td>3.83931</td>
<td>3.017138</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>4.35590</td>
<td>3.810338</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cray T3E-300
# Results: Relative Speedups and Overheads

## Relative Speedup for variable P and N/P ratio (without data packing)

<table>
<thead>
<tr>
<th>n/P</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7.16312</td>
<td>13.70352</td>
<td>23.75986</td>
<td>29.32111</td>
</tr>
<tr>
<td>4</td>
<td>7.65047</td>
<td>14.67344</td>
<td>25.45962</td>
<td>35.79221</td>
</tr>
<tr>
<td>8</td>
<td>10.19116</td>
<td>17.22411</td>
<td>34.84956</td>
<td>37.75083</td>
</tr>
<tr>
<td>16</td>
<td>10.07525</td>
<td>16.90213</td>
<td>31.05856</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>10.76393</td>
<td>19.32016</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>10.77978</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Relative Speedup for variable P and n/P ratio (with data packing)

<table>
<thead>
<tr>
<th>n/P</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7.77979</td>
<td>14.93582</td>
<td>25.51824</td>
<td>30.78324</td>
</tr>
<tr>
<td>4</td>
<td>8.35691</td>
<td>15.95337</td>
<td>27.6817</td>
<td>37.8441</td>
</tr>
<tr>
<td>8</td>
<td>11.25773</td>
<td>19.04754</td>
<td>38.52409</td>
<td>40.34156</td>
</tr>
<tr>
<td>16</td>
<td>11.08712</td>
<td>18.69831</td>
<td>34.25096</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>11.8361</td>
<td>21.40004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>11.8034</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Speedups (with data packing)

<table>
<thead>
<tr>
<th>n/P</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8.61%</td>
<td>8.99%</td>
<td>7.40%</td>
<td>4.99%</td>
</tr>
<tr>
<td>4</td>
<td>9.23%</td>
<td>8.72%</td>
<td>8.73%</td>
<td>5.73%</td>
</tr>
<tr>
<td>8</td>
<td>10.47%</td>
<td>10.59%</td>
<td>10.54%</td>
<td>6.86%</td>
</tr>
<tr>
<td>16</td>
<td>10.04%</td>
<td>10.63%</td>
<td>10.28%</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>9.96%</td>
<td>10.77%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>9.50%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Overheads (communication, synchronization, etc.) for variable n/P

<table>
<thead>
<tr>
<th>n/P</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.008265</td>
<td>0.009625</td>
<td>0.012381</td>
<td>0.019926</td>
<td>0.041543</td>
</tr>
<tr>
<td>4</td>
<td>0.013282</td>
<td>0.020122</td>
<td>0.027627</td>
<td>0.037403</td>
<td>0.033090</td>
</tr>
<tr>
<td>8</td>
<td>0.02527</td>
<td>0.026691</td>
<td>0.042316</td>
<td>0.056722</td>
<td>0.136465</td>
</tr>
<tr>
<td>16</td>
<td>0.038318</td>
<td>0.057994</td>
<td>0.104246</td>
<td>0.121884</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>0.079183</td>
<td>0.105193</td>
<td>0.149587</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>0.172304</td>
<td>0.220757</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© Andrés Jaramillo-Botero, 2003
Molecular Animations
Concluding Remarks

☑ Strictly parallel, minimum computational complexity algorithm of $O(\log_2 n)$ using $O(n)$ processors for the solution to the equations of motion (inverse and forward problems) in large scale systems.

☑ For $n > P$ systems computational complexity remains optimum at $O(\log_2 P + n/P)$ using $O(n)$ processors.

☑ Scalability demonstrated for certain systems and Speedups of up to 60%P-fold.

☑ Effective Increase of $\Delta t$’s using rigid body molecular dynamics (10 times conventional - 30fs).

☑ Solutions to the dynamics problems (inverse and forward) of articulated multi-bodies applicable in other fields (e.g. Spatial robotics, control).

☑ Hyper-branched and closed chain solutions presented, again with minimum computational complexity for strictly parallel implementations. $O(\log_2 N + l)$ for $l$ bodies per branch using $O(n)$ processors, and $O(\log_2 n)$ using $O(2n)$ processors, respectively.

☑ Added improvements can include software pipelining for inverse dynamics problem solution, support for $> 1$ DOF joints given the inverse mass operator is given as a function of degrees of constraint matrices.

☑ Variant of the cyclic reduction scheme for block matrix factorization and inversion ($N = P$), resulted in increased performance over conventional parallel method.

☑ Considerable grain size difference between forward and inverse solutions to the dynamics problem. Latency costs still a major cost in high grain parallelism.
Publications


© Andrés Jaramillo-Botero, 2003
Acknowledgements

- Dr. Amir Fijany, JPL, US
- Dr. Tahir Cagin, MSC Caltech, US
- Dr. Mario Blanco, MSC Caltech, US
- Dr. William Goddard III, MSC Caltech, US
- Pontificia Universidad Javeriana – Cali, Colombia
- Materials and Process Simulation Center (MSC-Caltech)
- Jet Propulsion Laboratory (JPL, Engineering Directorate)
- Universidad Politécnica de Valencia, Spain