CS 598: Communication Cost Analysis of Algorithms
Lecture 24: Molecular dynamics

Edgar Solomonik

University of Illinois at Urbana-Champaign

November 14, 2016

Particle simulation

There are many types of N-body simulations

- generally they involve N interacting particles simulated over time
- molecular dynamics and cosmological simulations are particularly important
    - many types of methods exist for both
    - can be simulated directly by calculating all N^2 pairwise interactions
    - a key difference is the distribution of particles versus the distribution of planets and stars
- most numerical methods take advantage of the decay of the strength of interactions with distance
- first we consider direct interaction calculations, then methods that compute interactions within a cutoff distance

Molecular dynamics (MD) high-level schematic

A molecular dynamics simulation performs the following calculations at every *timestep*

1. calculate non-bonded forces $F(i, j)$ for each pair of particles $p(i)$, $p(j)$
2. integrate non-bonded forces $f(i) = \sum_j F(i, j)$
3. consider local bonded many-particle interactions and update $f(i)$
4. update acceleration $a(i) = f(i)/m(i)$ and velocity $v(i)$ using $a(i)$
5. compute new particle position $x(i)$ using $v(i)$ and $a(i)$

Example force potential

In classical MD simulation, there two key types of non-bonded forces

- Van der Waals (dipole) interactions
  - refer to local particle interactions
  - are generally approximations to the electronic wavefunction
  - a popular simple formulation is the Lennart-Jones potential

$$F_{\mathrm{LJ}}(i,j) = \frac{1}{x(i)-x(j)}\left(\frac{\sigma_{ij}^{(A)}}{|x(i)-x(j)|^{12}} - \frac{\sigma_{ij}^{(B)}}{|x(i)-x(j)|^{6}}\right)$$

  where $\sigma_{ij}^{(A)}$ and $\sigma_{ij}^{(B)}$ depend on the type of particle $p(i)$ and $p(j)$ are
- electrostatic interactions
  - described by Coulomb's law for electric field due to charge
  - decay slowly relative to Van Der Waals interactions

$$F_{\mathrm{EC}}(i,j) = (x(i)-x(j))\frac{q(i)q(j)}{|x(i)-x(j)|^{3}}$$

  where $q(i)$ and $q(j)$ are the charges of $p(i)$ and $p(j)$

Example force integration

There are different schemes for updating x and v from the acceleration a

- time is discretized, so it can make sense to take into account values of velocity and acceleration v_{old} and a_{old} from the previous iteration
- different schemes can preserve various quantities and may have different error
- *Velocity Verlet* is particularly common because it preserves total energy and has second order global error in the time-step size s
    1. $v(i) = v_{\text{old}}(i) + \frac{1}{2}(a_{\text{old}}(i) + a(i))s$
    2. $x(i) = x_{\text{old}}(i) + v_{\text{old}}(i)s + \frac{1}{2}a(i)s^2$

Cache complexity of direct interactions

First, lets consider the memory-cache traffic of local MD calculation

- Q: provided we can fit $\Theta(H)$ particles into cache, how much useful computation can be done with this set?
- A: $\Theta(H^2)$, outputting $\Theta(H)$ partial force calculations
- its possible to compute all N^2 interaction pairs with only $O(N^2/H)$ cache complexity

Particle decomposition

The simplest was to parallelize MD is *particle decomposition*

- each processor is assigned N/P particles
- processors exchange particles in a ring communicator, computing forces from received processors to their own N/P
- Q: what communication complexity does this scheme have?
- A: $O(N \cdot \beta + P \cdot \alpha)$
- its possible to have fewer messages when more memory is available

Force decomposition

Particle decomposition corresponds to a row-wise blocking of F

- alternatively we can have each processor calculate an $N/\sqrt{P} \times N/\sqrt{P}$ block of the force matrix F
- each processor would require $2N/\sqrt{P}$ particles $p(i)$ and $p(j)$ for each $F(i,j)$ it computes
- a reduction is necessary to compute $f(i) = \sum_j F(i,j)$
- total communication cost is $O(N/\sqrt{P} \cdot \beta + \alpha)$
- Q: what is the disadvantage of this method over particle decomposition?
- A: the memory usage is $\Theta(N/\sqrt{P})$ rather than $\Theta(N/P)$

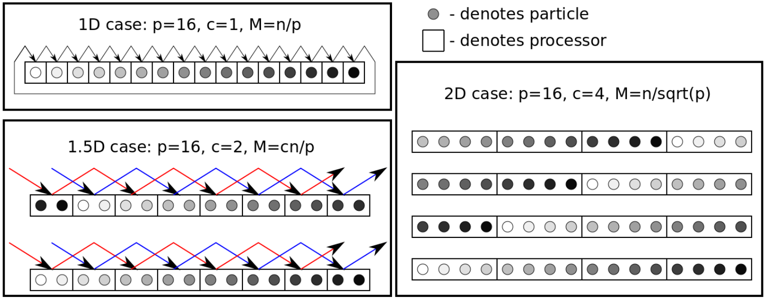
Memory-constrained force decomposition

We can trade off memory-usage and communication cost

- classical MD is often performed on relatively small systems (N) on a large number of processors (P) so as to simulate a sufficiently long time-period, so pure force decomposition is actually often acceptable
- in general, we may have $M = \Theta(cN/P)$ memory per processor for some $c \in [1, \sqrt{P}]$
- can apply same reasoning as for cache complexity:
  - with $\Theta(M)$ particles in memory can do $\Theta(M^2)$ useful work
  - so, if each processor computes $N^2/(PM^2)$ different $M \times M$ blocks of F

$$\begin{aligned} T_{\text{MF}}(N, P, M) &= O\Big(\frac{N^2}{P} \cdot \gamma + \frac{N^2}{PM} \cdot \beta + \frac{N^2}{PM^2} \cdot \alpha\Big) \\ &= O\Big(\frac{N^2}{P} \cdot \gamma + \frac{N}{c} \cdot \beta + \frac{P}{c^2} \cdot \alpha\Big) \end{aligned}$$

Algorithms for direct force calculation


1D case: p=16, c=1, M=n/p
1.5D case: p=16, c=2, M=cn/p
● - denotes particle
□ - denotes processor
2D case: p=16, c=4, M=n/sqrt(p)

- 1D – particle decomposition
- 2D – force decomposition
- 1.5D – memory-constrained force decomposition

Short pause

Cutoff radius

Few real applications actually calculate all particle interactions

- Van der Waals interactions decay very rapidly and can be ignored for far-away particles
- electrostatic forces can be computed by fast solvers
  - electrostatic potential obeys the Poisson equation
  - the gravitational potential (used for cosmological simulation) is also Poisson
- general structure of methods is as follows
  - compute Van der Waals interactions of all particles $p(i)$, $p(j)$ within distance $|x(i) - x(j)| \leq r_c$
  - construct a 3D charge density grid
  - solve the 3D Poisson equation on the grid via 3D FFT or Multigrid
  - interpolate potential to compute electrostatic forces

Parallel spatial decomposition

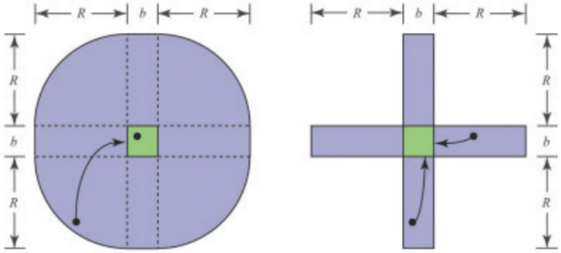
Let the domain be a $N^{1/3} \times N^{1/3} \times N^{1/3}$ box and assume uniform density

- molecular dynamics simulations are typically done inside ‘solute’ (water), and have uniform density
  - there are also *implicit methods*, which avoid working with water molecules explicitly, but they are less accurate and require more expensive interaction calculations
- cosmological simulations have highly non-uniform density
- if we assign each processor $\Theta(N/P)$ particles in a subdomain of dimensions $(N/P)^{1/3} \times (N/P)^{1/3} \times (N/P)^{1/3}$
  - to compute forces onto all these particles, need all particles within r_c away from subdomain
  - Q: how much communication does this require?
  - A: $O((r_c + (N/P)^{1/3})^3 - N/P) = O(r_c^3 + r_c(N/P)^{2/3})$

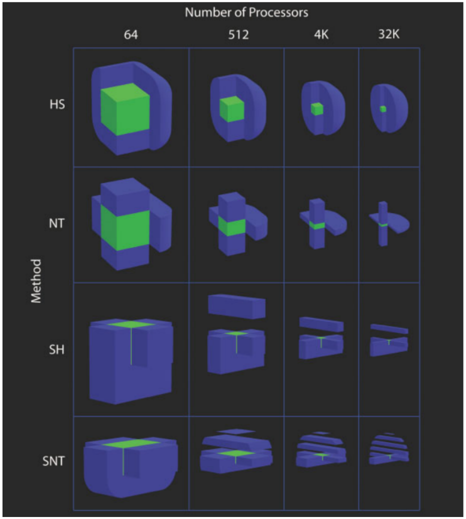
Neutral territory methods

An even better approach is to decompose the space of forces

- allow interactions between particles owned by two different processors to be computed on a third, in “neutral territory”
- David Shaw and Marc Snir came up with two important variants of these methods
- Shaw’s approach attains the communication complexity $O(r_c^{3/2}(N/P)^{1/2} + r_c(N/P)^{2/3})$
- in the 2D case it uses the following decomposition



Neutral territory methods



Diagrams taken from D. Shaw, “A Fast, Scalable Method for the Parallel Evaluation of Distance-Limited Pairwise Particle Interactions”, 2005

Neutral territory methods

In the NT method, each processor k is assigned a unique subvolume $X_k \times Y_k \times Z_k$ of dimensions $b_{xy} \times b_{xy} \times b_z$ such that $b_{xy}^2 b_z = N/P$

- it computes interactions of all particle $p(i)$ and $p(j)$ such that
  - $p(i)$ and $p(j)$ have a z-coordinate in Z_k and x, y-coordinates within r_c of some element in X_k, Y_k, respectively
  - $p(i)$ and $p(j)$ have x, y-coordinates in X_k, Y_k and a $z-$coordinate within r_c of some element in Z_k
- the volume of the region (the amount of communication) required is

$$W_{\mathrm{NT}}(r_c, b_{xy}, b_z) = O(r_c b_{xy}^2 + r_c b_z b_{xy} + r_c^2 b_z)$$

Neutral territory methods

We need to minimize

$$W_{\mathrm{NT}}(r_c, b_{xy}, b_z) = O(r_c b_{xy}^2 + r_c b_z b_{xy} + r_c^2 b_z)$$

- subject to $b_{xy}^2 b_z = N/P$
- note that $b_z = N/(P b_{xy}^2)$ so

$$W_{\mathrm{NT}}(r_c, b_{xy}, N, P) = O\left(r_c b_{xy}^2 + \frac{r_c N}{P b_{xy}} + \frac{r_c^2 N}{P b_{xy}^2}\right)$$

- minimizing this quantity gives

$$\min_{b_{xy}}(W_{\mathrm{NT}}(r_c, b_{xy}, N, P)) = \begin{cases} r_c < (N/P)^{1/3} : O(r_c (N/P)^{2/3}) \\ r_c \geq (N/P)^{1/3} : O(r_c \sqrt{r_c N/P}) \end{cases}$$