

Parallel Numerical Algorithms

Chapter 16 – Particle Simulations

Prof. Michael T. Heath

Department of Computer Science
University of Illinois at Urbana-Champaign

CSE 512 / CS 554

N-Body Model

- Newton's Second Law

$$F = m a$$

- Force between particles at positions x_i and x_j

$$f(x_i, x_j)$$

- Overall force on i th particle

$$F(x_i) = \sum_{j=1}^n f(x_i, x_j)$$

Reducing Cost of Force Evaluation

- Use Newton's Third Law: $f(x_i, x_j) = -f(x_j, x_i)$ to reduce work by essentially half
- Use cutoff radius R and update force due to particles more distant than R less often, thereby reducing cost of force evaluation to $\mathcal{O}(n R^3 + \epsilon n^2)$
- Constrain groups of particles to move together using, e.g., SHAKE algorithm
- Use hierarchical ("tree") or multipole methods to reduce cost to $\mathcal{O}(n \log n)$ or even $\mathcal{O}(n)$, but with some sacrifice in accuracy

Parallelizing Particle-Particle Method

- Arrange tasks in 2-D mesh, where task (i, j) computes force between particles i and j
- Let diagonal tasks be "home" to respective particles
- Each force pair computation is perfectly parallel

$$f_{11} \ f_{12} \ f_{13} \ f_{14} \ \dots \ f_{1n}$$

$$f_{21} \ f_{22} \ f_{23} \ f_{24} \ \dots \ f_{2n}$$

$$f_{n1} \ f_{n2} \ f_{n3} \ f_{n4} \ \dots \ f_{nn}$$

N-Body Problems

- Many physical systems can be modeled as collection of interacting particles
- "Particles" vary from atoms in molecule to planets in solar system or stars in galaxy
- Particles exert mutual *forces* on each other, such as gravitational or electrostatic forces

N-Body Simulation

- System of ODEs

$$F(x_i) = m_i \frac{d^2 x_i}{dt^2}$$

- For long time integration, *symplectic* integrators are appropriate, to preserve geometric properties
- Popular example is *Verlet* time-stepping scheme

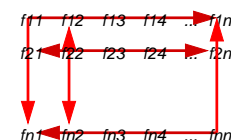
$$x_i^{k+1} = 2x_i^k - x_i^{k-1} + (\Delta t)^2 F(x_i^k)/m_i$$

- $\mathcal{O}(n^2)$ cost of evaluating force at each time step dominates overall computational cost

Parallel Particle Simulations

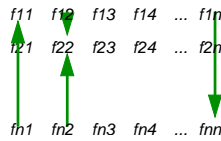
- Straightforward force evaluation is naturally parallel but total work is prohibitive and memory requirements may be excessive
- Methods for reducing total work also complicate parallel implementation

Particle-Particle Method



- Broadcast position of particle i to all tasks in same row and column

Particle-Particle Method



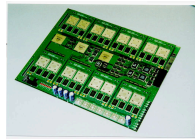
- Reduce forces to diagonal along column and perform time integration
- Due to symmetry, could reduce along rows instead

Implementing Broadcasts

- Simplest approach is to consider this an all-to-all operation: each process sends positions of particles that it owns to all other processes
- Use `MPI_Alltoall` if each process has same number of particles or `MPI_Alltoallv` otherwise
- This operation is very communication-intensive and must be completed before any computation
- In addition, each process must store locations of all n particles
- Is there an alternative?

Digital Orrery

- This algorithm is sometimes called a *digital orrery*
- Introduced in 1985 paper in IEEE TOC, 10 SIMD computers connected in a ring
- GRAPE computers extend approach of using special-purpose hardware, achieving over 2 PetaFLOPS sustained (using n^2 direct algorithm). See nbodylab.interconnect.com/nb1_grape_history.html

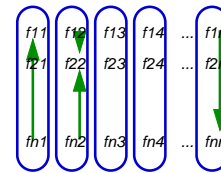


GRAPE-6 board

Handling Long-Range Forces

- Forces have infinite range, but with declining strength
- Available options thus include
 - Perform full computation at $\mathcal{O}(n^2)$ cost
 - Discard forces from particles beyond certain range, introducing error that is bounded away from zero
 - Approximate long-range forces, exploiting behavior of force and/or features of problem
- Various approaches can be taken to approximating long-range forces

Particle-Particle Method



- Agglomerate by *columns*, so reduction requires no communication — need to broadcast only by rows
- Due to symmetry, could agglomerate along rows instead

Pipelined All-to-All

- Rather than perform All-to-all communication as single step, arrange communication in pipeline: first, process i sends locations of its particles to process $i + 1 \pmod p$ and receives from process $i + p - 1 \pmod p$
- Each process uses information received in computing force, then can discard position data
- Computation and communication can be overlapped
- Pipeline requires $p - 1$ steps, however, so algorithm is not scalable

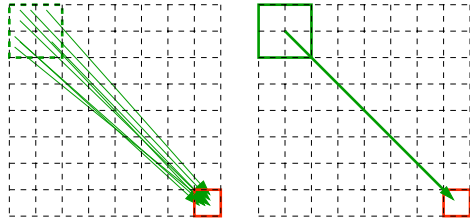
Improving Particle Algorithm

- Simple algorithm has two major drawbacks
 - work is $\mathcal{O}(n^2)$
 - communication is $\mathcal{O}(p)$
- Reducing work may also allow us to reduce communication!

Monopole Representation

- Aggregate distant particles into cells and represent effect of all particles in cell by *monopole* (first term in multipole expansion) evaluated at center of cell
- Use larger cells at greater distances
- Leads to $\mathcal{O}(n \log n)$ cost
- But approximation is relatively crude
- Early version of this approach, often called *tree code*, was developed by Barnes and Hut

Tree Code for N-Body Problem



- Tree code approach replaces influence of each far-away particle with aggregate approximate force
- Shown here is replacement of forces from four boxes in upper left with one force used by box in lower right

Multipole Representation

- To avoid inaccuracy of monopole expansion, use full multipole expansion
- Simple approach provides $\mathcal{O}(n \log n)$ method with controllable accuracy, which can be better than direct method for large n due to reduced roundoff error
- Additional tricks allow collecting terms, reducing complexity to $\mathcal{O}(n)$, but with substantial constant
- This is **Fast Multipole Method** of Greengard and Rokhlin

Limitations of Particle-in-Cell

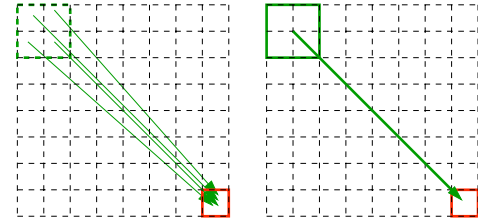
- Smoothing out particles introduces significant error
- Error may be reduced (but not eliminated) by splitting force into two parts,

$$F = F_{near} + F_{far}$$
- Compute force due to far-away particles, F_{far} , using particle-in-cell, removing contributions from nearby particles
- Compute force due to near-by particles, F_{near} , using simple particle-particle method
- This is known as particle-particle-particle-in-mesh, or PPPM

Final Remarks

- Methods for n-body problem often trade accuracy for work
- Success often depends critically on time integration scheme and model of forces
- Load balancing can be critical for scalable performance
- Use of task or process *virtualization* can help organize code
- Parallel approach may consider decompositions in space (as in tree and multipole methods) or particles (as in digital orrery algorithm)

Parallelizing Tree Code



- Divide up domain into patches, with each process assigned to a patch
- Tree code replaces communication with all processes by communication with fewer processes

Particle-in-Cell

- For many n -body calculations, force can be represented as gradient of potential, or $F = -\nabla\phi$
- Potential ϕ is related to forces produced by particles through **field equation**

$$\nabla^2\phi = -c\rho,$$

- where ρ is charge for electostatics or mass density for gravity
- This suggests simple approach: define mesh and assign particles to nodes on mesh, preserving charge or mass
- Solve Poisson problem then compute force as $F = -\nabla\phi$

Parallelizing Particle-in-Cell

- We already know how to solve Poisson problem in parallel, for example using FFT or multigrid
 - FFT requires significant communication
 - Multigrid reduces communication requirements and may scale better
- Load balancing requires more adaptive approach to assigning of particles to processes

References

- M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*, Oxford University Press, 1987
- D. Frenkel and B. Smit, *Understanding Molecular Simulation: From Algorithms to Applications*, 2nd ed., Academic Press, 2002
- J. M. Haile, *Molecular Dynamics Simulations: Elementary Methods*, Wiley, 1992
- R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, Institute of Physics, 1988
- B. Leimkuhler and S. Reich, *Simulating Hamiltonian Dynamics*, Cambridge University Press, 2005

References

- J. A. McCammon, B. M. Pettitt, and L. R. Scott, Ordinary differential equations of molecular dynamics, *Comput. Math. Appl.*, 28:319-326, 1994
- S. Pflanzner and P. Gibbon, *Many-Body Tree Methods in Physics*, Cambridge University Press, 1996
- S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, *J. Comput. Physics* 103:3668-3679, 1995
- D. C. Rapaport, *The Art of Molecular Dynamics Simulation*, Cambridge University Press, 1995
- T. Schlick, *Molecular Modeling and Simulation: An Interdisciplinary Guide*, Springer, 2002

