Parallel Numerical Algorithms

Chapter 16 – Particle Simulations

Prof. Michael T. Heath

Department of Computer Science
University of Illinois at Urbana-Champaign

CS 554 / CSE 512

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 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 ith particle

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

N-Body Simulation

- System of ODEs

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

- *Verlet* time-stepping scheme

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

- For long time integration, *symplectic* integrators are appropriate (preserve geometric properties, such as orbits)
- $\mathcal{O}(n^2)$ cost of evaluating force at each time step dominates overall computational cost

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

Parallel Particle Simulations

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

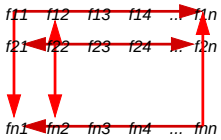
Parallelizing Particle-Particle Method

- Arrange tasks in 2-D grid, with task (i, j) computing force between particles i and j
- Let diagonal elements be "home" to respective particles
- Each force pair computation is perfectly parallel:

f11 f12 f13 f14 ... f1n
f21 f22 f23 f24 ... f2n

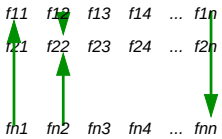
fn1 fn2 fn3 fn4 ... fnn

Particle-Particle Method



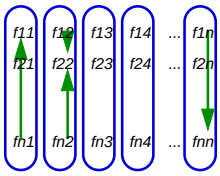
- Broadcast position of particle i to all tasks in same row and column

Particle-Particle Method



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

Particle-Particle Method



- If agglomerate by *columns*, then reduction requires no communication, and broadcast needed only across rows
- Due to symmetry, could agglomerate along rows instead

Implementing Broadcasts

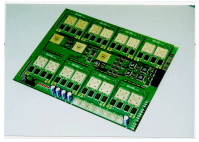
- Simplest approach is to consider this an all-to-all operation: each process sends positions of particles that it owns to all other processes
- Use `MPI_Alltoall` if each process has same number of particles or `MPI_Alltoallv` otherwise
- This is very communication-intensive and must be completed before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

Pipelined All-to-All

- Rather than perform All-to-all communication as single step, arrange communication in pipeline: in first step, 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 the position data
- Computation and communication can be overlapped
- Pipeline requires $p-1$ steps, however, so algorithm is not scalable

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/nbl_grape_history.html`



GRAPE-6 board

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 reduction in communication

Handling Long Range Forces

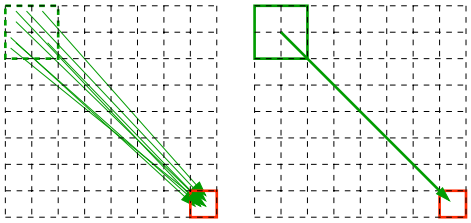
- Forces have infinite range, but with declining strength
- Three major options
  - Perform full computation at $\mathcal{O}(n^2)$ cost
  - Discard forces from particles beyond certain range, introducing error that is bounded away from zero
  - Approximate long-range forces, exploiting behavior of force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

Monopole Representation

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

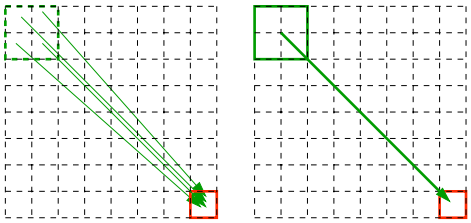
Tree Code for N-Body Problems

- 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

Parallelizing Tree Code



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

Multipole Representation

- To avoid accuracy problem of monopole expansion, use full multipole expansion
- Simple approach yields $\mathcal{O}(n \log n)$ method with controllable accuracy; can be more accurate than direct method for large n due to reduced rounding error
- Additional tricks allow collecting terms, reducing complexity to $\mathcal{O}(n)$ (but with substantial constant), giving *Fast Multipole Method* of Greengard and Rokhlin

Particle-in-Cell

- For many n-body calculations, force can be represented as gradient of a potential:

$$F = -\nabla\phi$$

- Potential ϕ related to forces produced by particles through *field equation*
- For electostatics or gravity,

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

where ρ is charge or mass density

- This suggests simple approach: Define mesh and assign particles to nodes of mesh, preserving charge or mass
- Solve Poisson problem
- Compute force as $F = -\nabla\phi$

Limitations of Particle-in-Cell

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

$$F = F_{near} + F_{far}$$

- Compute F_{far} force due to far-away particles using Particle-in-cell; must remove contribution from nearby particles
- Compute F_{near} force due to near-by particles using simple particle-particle method
- Known as particle-particle-particle-in-mesh, or PPPM, method

Parallelizing Particle in Cell Algorithm

- Already know how to solve Poisson problem in parallel using methods such as FFT or multigrid
- FFT requires significant communication
- Multigrid reduces communication requirements and may scale better
- Load balancing requires more adaptive approach to assignment of particles to processes

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 crucial to achieve scalable performance
- Task or process *virtualization* can help organize code
- Parallel approach may consider decompositions in space (as in tree or multipole methods) or particles (as in digital orrery)

References - Particle Simulations

- 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
- M. Griebel, S. Knapek, and G. Zumbusch, *Numerical Simulation in Molecular Dynamics: Numerics, Algorithms, Parallelization, Applications*, Springer, 2007
- J. M. Haile, *Molecular Dynamics Simulations: Elementary Methods*, Wiley, 1992
- E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, 2nd ed., Springer, 2006

References - Particle Simulations

- 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
- J. A. McCammon, B. M. Pettitt, and L. R. Scott, Ordinary differential equations of molecular dynamics, *Comput. Math. Appl.*, 28:319-326, 1994
- S. Pfalzner and P. Gibbon, *Many-Body Tree Methods in Physics*, Cambridge University Press, 1996
- D. C. Rapaport, *The Art of Molecular Dynamics Simulation*, Cambridge University Press, 1995
- T. Schlick, *Molecular Modeling and Simulation: An Interdisciplinary Guide*, 2nd ed., Springer, 2010

References - Parallel Particle Simulations

- M. Driscoll et al., A communication-optimal n-body algorithm for direct interactions, IPDPS, Boston, May 2013
- B. A. Hendrickson and S. J. Plimpton, Parallel many-body simulations without all-to-all communication, *J. Parallel Distrib. Comput.* 27:15-25, 1995
- S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, *J. Comput. Physics* 117:1-19, 1995
- H. Schreiber, O. Steinhauser, and P. Schuster, Parallel molecular dynamics of biomolecules, *Parallel Comput.* 18:557-573, 1992
- W. Smith, Molecular dynamics on hypercube parallel computers, *Comp. Phys. Comm.* 62:229-248, 1991
- M. Snir, A note on n-body computations with cutoffs, *Theory Comput. Sys.* 37:295-318, 2004