CS601: Software Development for Scientific Computing Autumn 2024

Week15 : N-Body Problems

Course Progress..

• Pic source: the Parallel Computing Laboratory at U.C. Berkeley: A Research Agenda Based on the Berkeley View (2008)



Figure 4. Temperature Chart of the 13 Motifs. It shows their importance to each of the original six application areas and then how important each one is to the five compelling applications of Section 3.1. More details on the motifs can be found in (Asanovic, Bodik et al. 2006).



Particle (Simulation) Methods

• N-Body Simulation – Problem

System of N-bodies (e.g. galaxies, stars, atoms, light rays etc.) interacting with each other continuously

- Problem:
 - Compute force acting on a body due to all other bodies in the system
 - Determine position, velocity, at various times for each body
- Objective:
 - Determine the (approximate) evolution of a system of bodies interacting with each other simultaneously

Particle (Simulation) Methods

- N-Body Simulation Examples
 - Astrophysical simulation: E.g. each body is a star/galaxy <u>https://commons.wikimedia.org/w/index.php?title=File</u> <u>%3AGalaxy_collision.ogv</u>
 - Graphics: E.g. each body is a ray of light emanating from the light source.

https://www.fxguide.com/fxfeatured/brave-new-hair/

Here each body is a point on a strand of hair

N-Body Simulation

- All-pairs Method
 - Naïve approach. Compute all pair-wise interactions
- Hierarchical Methods
 - Optimize. Reduce the number of pair-wise force calculations. How? dependence on 'distant' particle(s) can be *compressed*
 - Examples:
 - Barnes-Hut
 - Fast Multipole Method

N-Body Simulation

- Three fundamental simulation approaches
 - Particle-Particle (PP)
 - Particle-Mesh (PM)
 - Particle-Particle-Mesh (P3M)
- Hybrid approaches
 - Nested Grid Particle Scheme
 - Tree Codes
 - Tree Code Particle Mesh (TPM)
- Self Consistent Field (SCF), Smoothed-Particle Hydrodynamics (SPH), Symplectic etc.

- Simplest. Adopts an all-pairs approach.
- State of the system at time t given by particle positions x_i(t) and velocity v_i(t) for i=1 to N

$$\{x_i(t), v_i(t); i = 1, N\}$$

- Steps:
 - → 1. Compute forces
 - 2. Integrate equations of motion
 - 3. Update time counter Each iteration updates $x_i(t)$ and $v_i(t)$ to compute $x_i(t + \Delta t)$ and $v_i(t + \Delta t)$

1. Compute forces

//initialize forces
for i=1 to N
 F_i = 0

//Accumulate forces
for i=1 to N-1
 for j=i+1 to N

$$F_i = F_i + F_{ij} \longleftarrow F_{ij}$$
 is the force on particle i due to particle j
 $F_j = F_j - F_{ij}$

Typically: $F_i = F_{external} + F_{nearest_neighbor} + F_{N-Body}$

2. Integrate equations of motion

for i=1 to N

$$v_i^{new} = v_i^{old} + \frac{F_i}{m_i} \Delta t$$
 //using a=F/m and v=u+at
 $x_i^{new} = x_i^{old} + v_i \Delta t$

3. Update time counter

$$t^{new} = t^{old} + \Delta t$$

Nikhil Hegde

```
t=0
while(t<t<sup>final</sup>) {
//initialize forces
         for i=1 to N
           F_i = 0
//Accumulate forces
         for i=1 to N-1
           for j=i+1 to N
              F[i] = F[i] + F_{ii}
              F[j] = F[j] - F_{ii}
//Integrate equations of motion
         for i=1 to N
           v_i^{new} = v_i^{old} + \frac{F_i}{m_i} \Delta t //using a=F/m and v=u+at
           x_i^{new} = x_i^{old} + v_i \Delta t
// Update time counter
        t = t + \Delta t
                                                                10
```

Costs (CPU operations)?



- Experimental results (then):
 - Intel Delta = 1992 supercomputer, 512 Intel i860s
 - 17 million particles, 600 time steps, 24 hours elapsed time

M. Warren and J. Salmon

Gordon Bell Prize at Supercomputing 1992

- Sustained 5.2 Gigaflops = 44K Flops/particle/time step
- 1% accuracy
- Direct method (17 Flops/particle/time step) at 5.2 Gflops would have taken 18 years, 6570 times longer

• Experimental results (now):

Vortex particle simulation of turbulence

- Cluster of 256 NVIDIA GeForce 8800 GPUs
- 16.8 million particles
 - T. Hamada, R. Yokota, K. Nitadori. T. Narumi, K. Yasoki et al
 - Gordon Bell Prize for Price/Performance at Supercomputing 2009
- Sustained 20 Teraflops, or \$8/Gigaflop

13

Discussion

- Simple/trivial to program
- High computational cost
 - Useful when number of particles are small (few thousands) and
 - We are interested in close-range dynamics when the particles in the range contribute *significantly* to forces
 - Constant time step must be replaced with variable time steps and numerical integration schemes for close-range interactions

N-Body Simulation

- All-pairs Method
 - Naïve approach. Compute all pair-wise interactions
- Hierarchical Methods
 - Optimize. Reduce the number of pair-wise force calculations. How? dependence on 'distant' particle(s) can be *compressed*
 - Examples:
 - Barnes-Hut
 - Fast Multipole Method

Tree Codes

- $F_i = F_{external} + F_{nearest_neighbor} + F_{N-Body}$ • $F_{external}$ can be computed for each body independently. O(N)
- $F_{nearest_neighbor}$ involve computations corresponding to few nearest neighbors. O(N)
- F_{N-Body} require all-to-all computations. Most expensive. O(N²) if computed using all-pairs approach.

for(i = 1 to N)

 $F_i = \sum_{i \neq j} F_{ij}$ $F_{ij} =$ force on i from j

 $F_{ij} = c^* v/||v||^3$ in 3D, $F_{ij} = c^* v/||v||^2$ in 2D v = vector from particle i to particle j , ||v|| =length of v, c = product of masses or charges

We can do better.

Tree Codes: Divide-Conquer Approach

- Consider computing force on earth due to all celestial bodies
 - > Look at the night sky. Number of terms in $\sum_{i \neq j} F_{ij}$ is greater than the number of visible stars
 - One "star" could really be the Andromeda galaxy, which contains billions of real stars. Seems like a lot more work than we thought ...
 - Idea: Ok to approximate all stars in Andromeda by a single point at its center of mass (CM) with same total mass (TM)



 \mathbf{x} = location of center of mass

- Require that D/r be "small enough" (D = size of box containing Andromeda, r

= distance of CM to Earth).

Idea is not new. Newton approximated earth and falling apple by CM

Slide contents based on: CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

Tree Codes: Divide-Conquer Approach

- New idea: recursively divide the box.
- If you are in Andromeda, Milky Way (the galaxy we are part of) could appear like a white dot. So, can be approximated by a point mass.
- Within Andromeda, picture repeats itself
 - As long as D1/r1 is small enough, stars inside smaller box can be replaced by their CM to compute the force on Vulcan
 - If you are on Vulcan, another solar system in Andromeda can be a white dot.
 - Boxes nest in boxes recursively

Nikhil Hegde



Replacing Clusters by their Centers of Mass Recursively

26

Slide contents based on: CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

Tree Codes: Divide-Conquer Approach

- Data structures needed:
 - Quad-trees
 - Octrees

Background – metric trees

e.g. K-dimensional (kd-), Vantage Point (vp-), quad-trees, octrees, balltrees

2-dimensional space of points



Binary kd-tree, 1 point /leaf cell



Nikhil Hegde

Background - metric trees

Typical use: traverse the tree (often repeatedly), truncate the traversal at some intermediate node if <u>a domain-</u> <u>specific criteria</u> is not met. E.g. Does the distance



E.g. Does the distance from CM to me < D/r?

Quad Tree

- Data structure to subdivide the plane
 - Nodes can contain coordinates of center of box, side length.
 - Eventually also coordinates of CM, total mass, etc.
- In a complete quad tree, each non-leaf node has 4 children



A Complete Quadtree with 4 Levels

Nikhil Hegde

Slide courtesy: CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

Octree or Oct Tree

• Similar data structure for subdividing 3D space

2 Levels of an Octree





Nikhil Hegde

31

Slide courtesy: CS267 Lecture 24, <u>https://sites.google.com/lbl.gov/cs267-spr2019/</u>

Using Quad Tree and Octree

- Begin by constructing a tree to hold all the particles
 - Interesting cases have nonuniformly distributed particles
 - In a complete tree most nodes would be empty, a waste of space and time
 - Adaptive Quad (Oct) Tree only subdivides space where particles are located
- For each particle, traverse the tree to compute force on it

Nikhil Hegde

Using Quad Tree and Octree

Adaptive quadtree where no square contains more than 1 particle



 In practice, have q>1 particles/square; tuning parameter

Slide courtesy: CS267 Lecture 24, <u>https://sites.google.com/lbl.gov/cs267-spr2019/</u>

Adaptive Quad Tree





• Child nodes numbered as per *Z*-order numbering

Adaptive Quad Tree Construction

Procedure Quad_Tree_Build Quad_Tree = {emtpy}

for j = 1 to N

... loop over all N particles

 Quad_Tree_Insert(j, root)
 ... insert particle j in QuadTree

endfor

- ... At this point, each leaf of Quad_Tree will have 0 or 1 particles
- ... There will be 0 particles when some sibling has 1

Traverse the Quad_Tree eliminating empty leaves ... via, say Breadth First Search

Procedure Quad_Tree_Insert(j, n) ... Try to insert particle j at node n in Quad_Tree if n an internal node n has 4 children

- determine which child c of node n contains particle j
- Quad_Tree_Insert(j, c)_

else if n contains 1 particle ... n is a leaf

- add n's 4 children to the Quad_Tree
- move the particle already in n into the child containing it
- let c be the child of n containing j
- Quad_Tree_Insert(j, c)

else

... n empty

```
- store particle j in node n
```

Nikhienegde

Slide courtesy: CS267 Lecture 24, <u>https://sites.google.com/lbl.gov/cs267-spr2019/</u>

Adaptive Quad Tree Construction – Cost?

Procedure Quad_Tree_Build		
Quad_Tree = {emtpy}	\leq N *max cost of Q	uad Tree Insert
for j = 1 to N	loop over all N particles	
Quad_Tree_Insert(j, root)	insert particle j in QuadTree	
endfor		

- ... At this point, each leaf of Quad_Tree will have 0 or 1 particles
- ... There will be 0 particles when some sibling has 1

Traverse the Quad_Tree eliminating empty leaves ... via, say Breadth First Search



Slide courtesy: CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

Adaptive Quad Tree Construction – Cost?

- Cost of calling Max Depth of Tree:
 - For uniformly distributed points?
 - For arbitrarily distributed points?
- Total Cost = ?

Adaptive Quad Tree Construction – Cost?

- Cost of calling Max Depth of Tree:
 - For uniformly distributed points? = $O(\log N)$
 - For arbitrarily distributed points? = O(b)
 - b is number bits used to represent the coordinates
- Total Cost = O(b N) or $O(N * \log N)$

Barnes-Hut

- Simplest hierarchical method for N-Body simulation
 - "A Hierarchical O(n log n) force calculation algorithm" by J. Barnes and P. Hut, Nature, v. 324, December 1986
- Widely used in astrophysics
- Accuracy $\geq 1\%$ (good when low accuracy is desired/acceptable. Often the case in astrophysics simulations.)

Barnes-Hut: Algorithm

(2D for simplicity)

- Build the QuadTree using QuadTreeBuild
 ... already described, cost = O(N log N) or O(b N)
- 2) For each node/subsquare in the QuadTree, compute the Center of Mass (CM) and total mass (TM) of all the particles it contains.
- 3) For each particle, traverse the QuadTree to compute the force on it,

Barnes-Hut: Algorithm (step 2)

Goal: Compute the Center of Mass (CM) and Total Mass (TM) of all the particles in each node of the QuadTree. (TM, CM) = Compute_Mass(root)

```
(TM, CM) = Compute Mass( n ) //compute the CM and TM of node n
  if n contains 1 particle
       //TM and CM are identical to the particle's mass and location
       store (TM, CM) at n
       return (TM, CM)
 else
    for each child c(j) of n //j = 1,2,3,4
          (TM(j), CM(j)) = Compute_Mass(c(j))
    endfor
    TM = TM(1) + TM(2) + TM(3) + TM(4)
    //the total mass is the sum of the children's masses
    CM = (TM(1)*CM(1) + TM(2)*CM(2) + TM(3)*CM(3) + TM(4)*CM(4)) / TM
    //the CM is the mass-weighted sum of the children's centers of mass
     store (TM, CM) at n
     return ( TM, CM )
 end if
                                                                  41
```

Slide based on : CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

Barnes-Hut: Algorithm (step 2 cost)

(2D for simplicity)

- Build the QuadTree using QuadTreeBuild
 ... already described, cost = O(N log N) or O(b N)
- 2) For each node/subsquare in the QuadTree, compute the Center of Mass (CM) and total mass (TM) of all the particles it contains.
 ... cost = O(number of nodes in the tree) = O(N log N) or O(b N)
- 3) For each particle, traverse the QuadTree to compute the force on it,

Barnes-Hut: Algorithm (step 3)

Goal: Compute the force on each particle by traversing the tree. For each particle, use as few nodes as possible to compute force, subject to accuracy constraint.

- For each node = square, can approximate force on particles outside the node due to particles inside node by using the node's CM and TM
- This will be accurate enough if the node is "far away enough" from the particle
- Need criterion to decide if a node is far enough from a particle
 - D = side length of node
 - r = distance from particle to CM of node
 - θ = user supplied error tolerance < 1
 - Use CM and TM to approximate force of node on box if D/r < θ



Slide based on : CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

Barnes-Hut: Algorithm (step 3)

//for each particle, traverse the QuadTree to compute the force on it for k = 1 to N

f(k) = TreeForce(k, root)

//compute force on particle k due to all particles inside root (except k)
endfor

function f = TreeForce(k, n)

//compute force on particle k due to all particles inside node n (except k)
f = 0

if n contains one particle (not k) //evaluate directly
 return f = force computed using direct formula

else

```
r = distance from particle k to CM of particles in n
```

D = side length of n

if D/r < q //ok to approximate by CM and TM
 return f = computed approximately using CM and TM</pre>

else

```
//need to look inside node
```

for each child c(j) of n //j=1,2,3,4

$$f = f + TreeForce (k, c(j))$$

end for

return f

end if





Nikhil Hegde





Nikhil Hegde



Nikhil Hegde

Barnes-Hut: Algorithm (step 3 cost)

- Correctness follows from recursive accumulation
 of force from each subtree
 - Each particle is accounted for exactly once, whether it is in a leaf or other node
- Complexity analysis
 - Cost of TreeForce(k, root) = O(depth of leaf containing k in the QuadTree)
 - Proof by Example (for $\theta > 1$):
 - For each undivided node = square, (except one containing k), $D/r < 1 < \theta$
 - There are at most 3 undivided nodes at each level of the QuadTree.
 - -There is O(1) work per node
 - -Cost = O(level of k)

Total cost = $O(\Sigma_k \text{ level of } k) = O(N \log N)$

Strongly depends on θ

Sample Barnes-Hut Force calculation For particle in lower right corner Assuming theta > 1



50

Slide based on : CS267 Lecture 24, <u>https://sites.google.com/lbl.gov/cs267-spr2019/</u>

Barnes-Hut: Algorithm (step 3 cost)

(2D for simplicity)

- Build the QuadTree using QuadTreeBuild
 ... already described, cost = O(N log N) or O(b N)
- 2) For each node/subsquare in the QuadTree, compute the Center of Mass (CM) and total mass (TM) of all the particles it contains.
 ... cost = O(number of nodes in the tree) = O(N log N) or O(b N)
- 3) For each particle, traverse the QuadTree to compute the force on it,
 ... cost depends on accuracy desired (θ) but still
 O(N log N) or O(bN)

Slide courtesy: CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

N-Body Simulation: Big Picture

• Recall:

```
t=0
while(t<t<sup>final</sup>) {
//initialize forces
```

```
//Accumulate forces
    BH(steps 1 to 3)
```

//Integrate equations of motion

```
//Update time counter
t = t + \Delta t
}
```

Fast Multipole Method (FMM)

- Can we make the complexity independent of the accuracy parameter (θ) ? FMM achieves this.
 - "Rapid Solution of Integral Equations of Classical Potential Theory", V. Rokhlin, J. Comp. Phys. v. 60, 1985 and
 - "A Fast Algorithm for Particle Simulations", L. Greengard and V. Rokhlin, J. Comp. Phys. v. 73, 1987.
- Similar to BH:
 - uses QuadTree and the divide-conquer paradigm
- Different from BH:
 - Uses more than TM and CM information in a box. So, computation is expensive and accurate than BH.
 - The number of boxes evaluated is fixed for a given accuracy parameter
 - Computes potential and not the Force as in BH

Background: Potential

- Force on a particle at (x, y, z) due to a particle at origin Proportional to $-\frac{(x,y,z)}{r^3}$ (This is called inverse-square law. Gravitational and electrostatic forces obey this.) where, $r = \sqrt{x^2 + y^2 + z^2}$
- Force is a vector. Potential is a scalar. Hence, potential is simpler to deal with.

Potential $\Phi(x, y, z) = -\frac{1}{r}$

• Negative of the gradient of potential = force

$$-\nabla\Phi(x, y, z) = -\left(\frac{d}{dx}\left(-\frac{1}{r}\right), \frac{d}{dy}\left(-\frac{1}{r}\right), \frac{d}{dz}\left(-\frac{1}{r}\right)\right)$$

Background: Potential

- In 2D, potential $\Phi(x, y) = \log r$
- Suppose we have N points (at $z_1, z_2, ..., z_N$, where $z_i = (x_i, y_i)$) in a plane with masses $m_1, m_2, ..., m_N$ resp.

then, their potential at
$$z = (x, y)$$
 is given by:

$$\Phi(x, y) = \sum_{i=1}^{N} m_i \log \left(\sqrt{(x - x_i)^2 + (y - y_i)^2} \right)$$

Goal: evaluate $\Phi(x, y)$ and its derivatives at N points $(z_1, z_2, ..., z_N)$ in O(N) time.

FMM Algorithm

- 1. Build the quadtree containing all the points.
- Traverse the quadtree from bottom to top, computing Outer(n) for each square n in the tree.
- 3. Traverse the quadtree from top to bottom, computing Inner(n) for each square in the tree.
- 4. For each leaf, add the contributions of nearest neighbors and particles in the leaf to Inner(n)

what is Outer(n) and Inner(n) ?

Well Separated Regions

 Compute the influence of all particles in source region (B) on every particle in target region (A)

(assumption: A and B are well-separated)



• At each point p_i in A, compute potential:

$$\Phi(x_i, y_i) = \sum_{p_j \in B} m_i \log |p_i - p_j|$$

i = 1 to N_A, j = 1 to N_B

• Cost: $O(N_A N_B)$ Nikhil Hegde

Well Separated Regions

Approximate the potential at every particle in target region

 (A) by the potential at C_A



• Cost: $O(N_A + N_B)$

Nikhil Hegde

Applying the 3-step Approximation

 In N-body simulation every point serves as source as well as target.

How to identify source and target (boxes A and B in previous slide) i.e. well-separated regions?

Hierarchical decomposition

• Level-0 decomposition

• Level-1 decomposition

А	В
С	D

Α

No well-separated boxes

• Level-2 decomposition

N1	N2	N3	A1
N4	В	N5	A2
N6	N7	N8	A3
A7	A6	A5	A4

Well-separated from B

Can approximate the influence of points in B on points in Ai s

What do we do about **B**'s influence on Ni s?

• Level-3 decomposition



Influence of points in Bi s on those in Ai s already computed at the previous level (level-2)

• Level-3 decomposition



Influence of points in Bi s on those in Ai s already computed at the previous level (level-2)

Well-separated from B4 Influence of B4's points on nx's points can be approximated

nx's constitute the <u>interaction list</u> for B4. What is the max size of interaction list? i.e. max number of nx s that we can have for any Bi?

Nikhil Hegde

• Level-3 decomposition



Influence of points in Bi s on those in Ai s already computed at the previous level (level-2)

Well-separated from B4 Influence of B4's points on nx's points can be approximated

What do we do about **B4**'s influence on its neighbors (white/unshaded boxes)?

• Level-4 decomposition



Any unshaded box outside can be the *target* for computing the influence of points in (*source*)

1. Computing Potential for Well-Separated Regions

- 1. for level L=2 to last_level
- 2. for each Box B at level L
- 3. iList = GetInteractionList(B)
- 4. for each well-separated box A in iList

//Compute potential

5. potential = $m_B \log |C_A - C_B|$

//Accumulate potential

 $\Phi(x_{C_A}, y_{C_A}) +=$ potential

Cost?

6.

1. Computing Potential for Well-Separated Regions

- 1. for level L=2 to last_level
- 2. for each Box B at level L
- 3. iList = GetInteractionList(B)
- 4. for each well-separated box A in iList

//Compute potential

5. potential = $m_B \log |C_A - C_B|$

//Accumulate potential

6. $\Phi(x_{C_A}, y_{C_A}) +=$ potential

Prereqs: we need m_B , C_A , C_B details. (step 0)

2. Assigning Potential to Points

1. for each Box A at level L=0 to last_level

2. $\Phi_{p_i} = \Phi_{p_i} + \Phi_{C_A}$ (where $p_i \in A$ and C_A is A's CM)

Cost?

3. Assigning Potential to Points (last level)

- 1. for each Box B at last_level
- 2. $\Phi_{p_i} = \Phi_{p_i} + \sum_{p_j \in Neighbors(B)} m_B \log |p_i p_j| \text{ (where } p_i \in B\text{)}$

Cost?

0. Computing Prereqs

- 1. for each Box B at level L=0 to last_level
- 2. $m_B = \sum_{p_j \in B} m_j$
- 3. //similarly compute C_B

Cost?

Total Cost (steps 0 + 1 + 2 + 3)

 $O(N \log N) + O(N) + O(N \log N) + O(N)$

Can we do better?