The Barnes-Hut N-Body Approximation

Ben Prather
UIUC Algorithms Interest Group,
Aug 3, 2016

Why Newtonian N-Body?

- For some important problems, GR is not a significant correction
  - Finite speed of gravity is not relevant
  - Bodies all have v << c, no gravitational waves
  - Curvature of space is small
  - Universal expansion can be added manually
- GR-based calculations become difficult for many-body interactions
- Can be coupled with fluid dynamics simulations, or field approaches (particle in cell)
- Also useful for Björk concerts

Problem: N-Body Simulation

- Compute accelerations of N bodies, each by summing the influence of the N-1 other bodies
- Each step is thus ½ * N * (N-1) ~ O(N^2) operations
  - But to realize ½ improvement adds ¼ * N * (N-1) memory requirement
- Time-domain advance via Euler, explicit RK, leapfrog, etc, etc.

$$\boldsymbol{a_n} = \sum_{i=1}^{N} \frac{G m_i}{r_{ni}^3} \boldsymbol{r_{ni}}$$

Why optimize?

- A galaxy contains ~10^{11} stars
- Blue Waters is ~10 Pflops = 10^{13} operations/sec
- At O(N^2): ~10^{22} operations
  - Step takes 10^9 seconds or ~30 years
- At O(N log(n)): ~10^{12} operations
  - Step takes 0.1 seconds!*

*Restrictions apply. See caveats later.

Barnes-Hut Overview

- Far-away masses pull in roughly the same direction

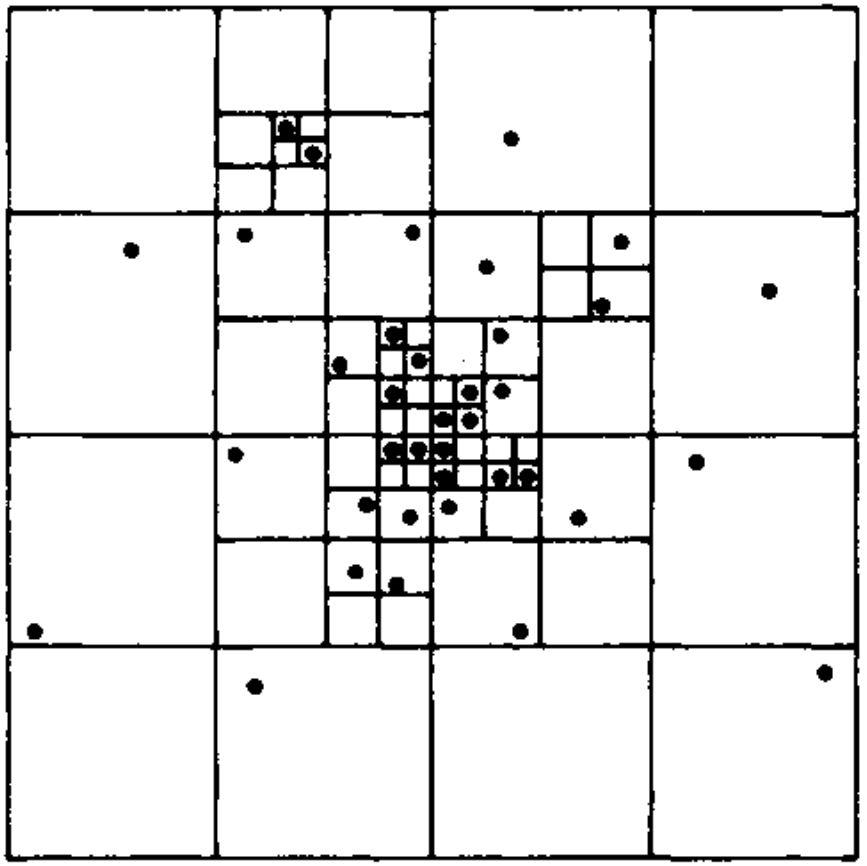
- Why not consolidate them into one?

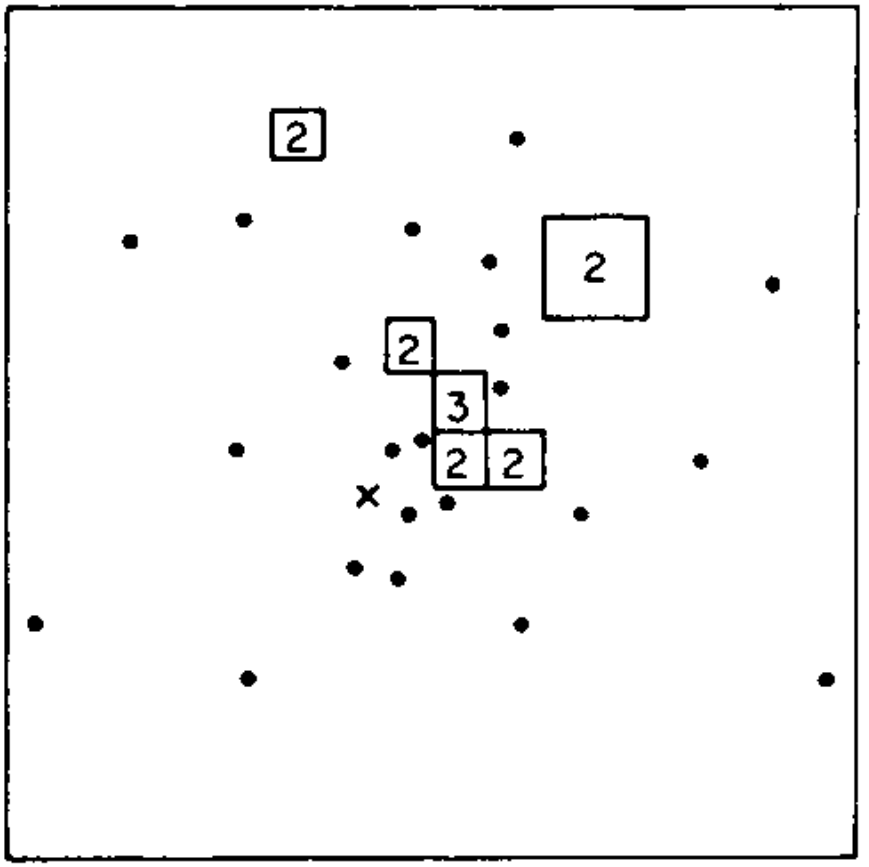
- Group by “cells.” Keep track of the center of mass and total mass of each cell
- Then, if a cell is small “enough” to consolidate, we can look up its CM and avoid touching all its children

Building the tree

- Add bodies to cells recursively to build a tree
- To add a body:
  - Start by adding the body to the root cell
  - Add the body to the cell's CM and mass
  - If the cell already encloses a body:
    - If the cell has no children, add them and propagate the body it currently encloses
    - Add the new body to the appropriate child and repeat if it is occupied
- Thus, there will be O(N) leaf cells always have one body at most, and a tree of larger cells
- Complexity is O(log N) per body, total O(N log(N))

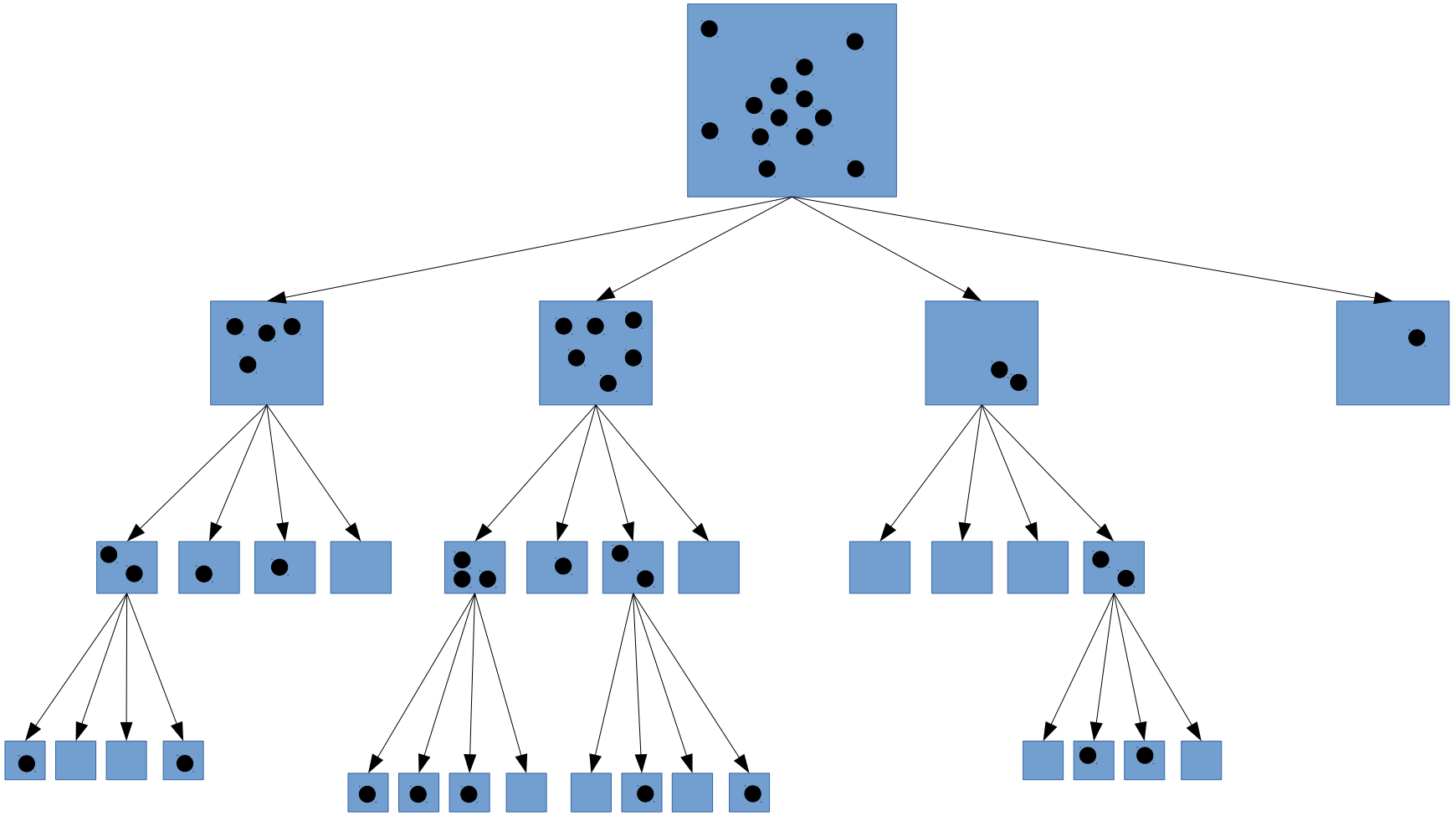
2D Tree Example



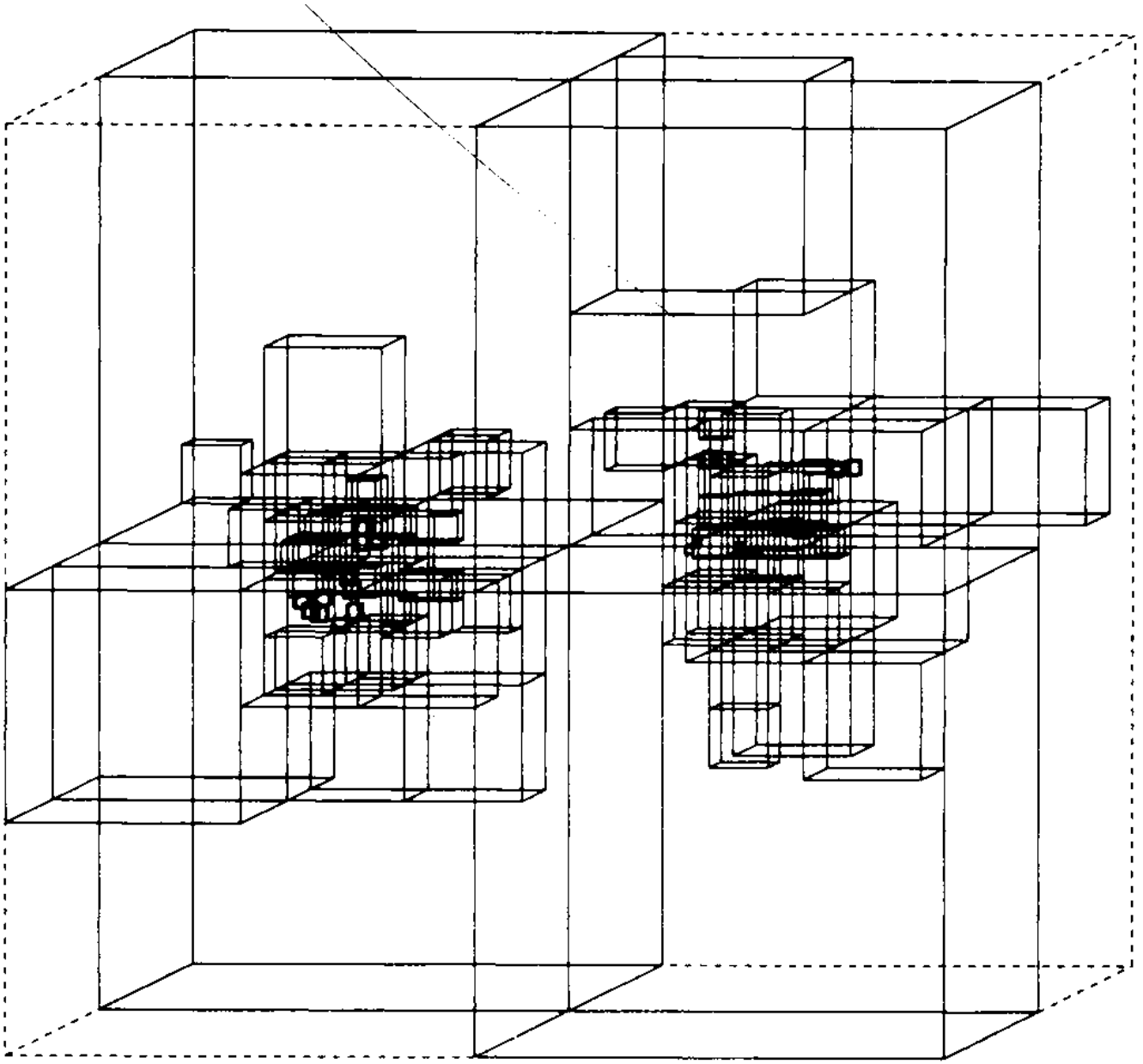


Images: Barnes & Hut

2D Tree in Memory



3D Tree



Step Algorithm

- To compute influence on a body:
  - Recursively walk the tree, summing the effect of child cells to obtain the effect of larger cells
  - When the diameter of the cell becomes less than a certain factor of its distance away, disregard its children and use its COM and mass
  - The total influence is the acceleration due to the root cell
- Only O(log N) cells have radii large enough to be summed

Step Algorithm: Details

- From the original paper (in LISP):

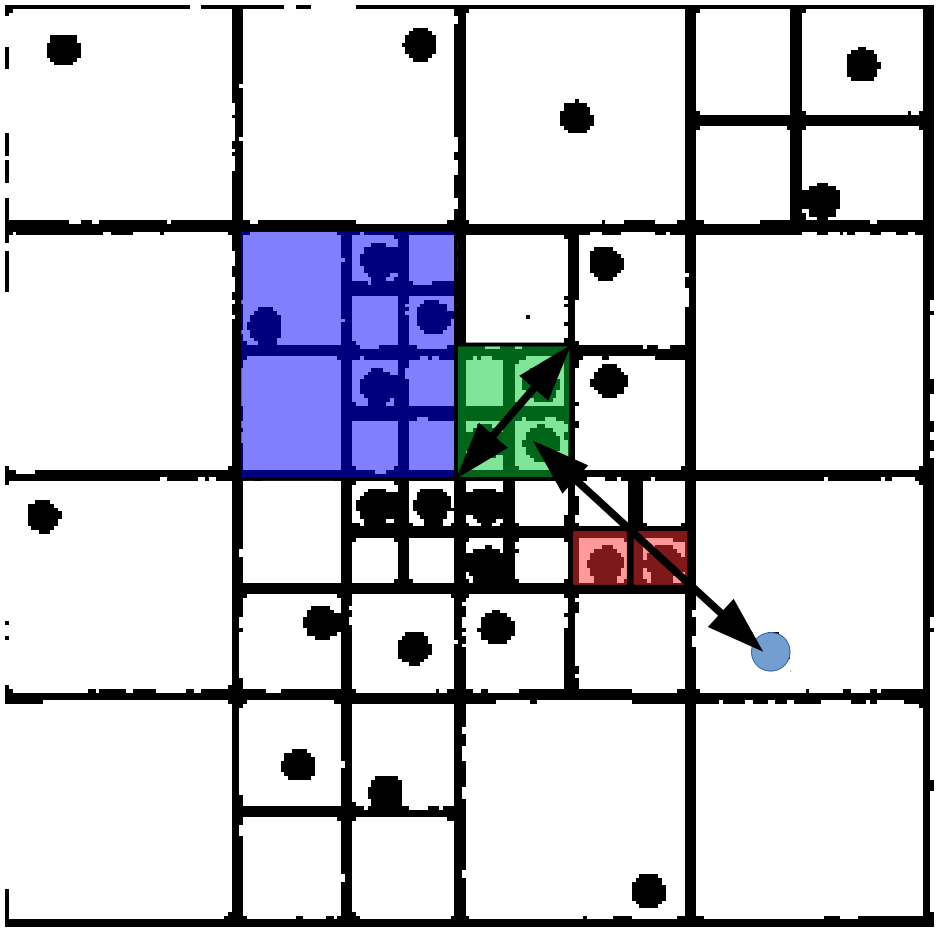
```
(define (acceleration particle ensemble)
   (cond ((singleton? ensemble)
          (newton-accel particle (the-element ensemble)))
         ((< (/ (diameter ensemble)
                (distance particle (centroid ensemble)))
            theta)
          (newton-accel particle (centroid ensemble)))
         (else
          (reduce sum-vector
               (map (lambda (e) (acceleration particle e))
                    (subdivisions ensemble))))))
```

Step Algorithm: Translation

- Perhaps more readable:

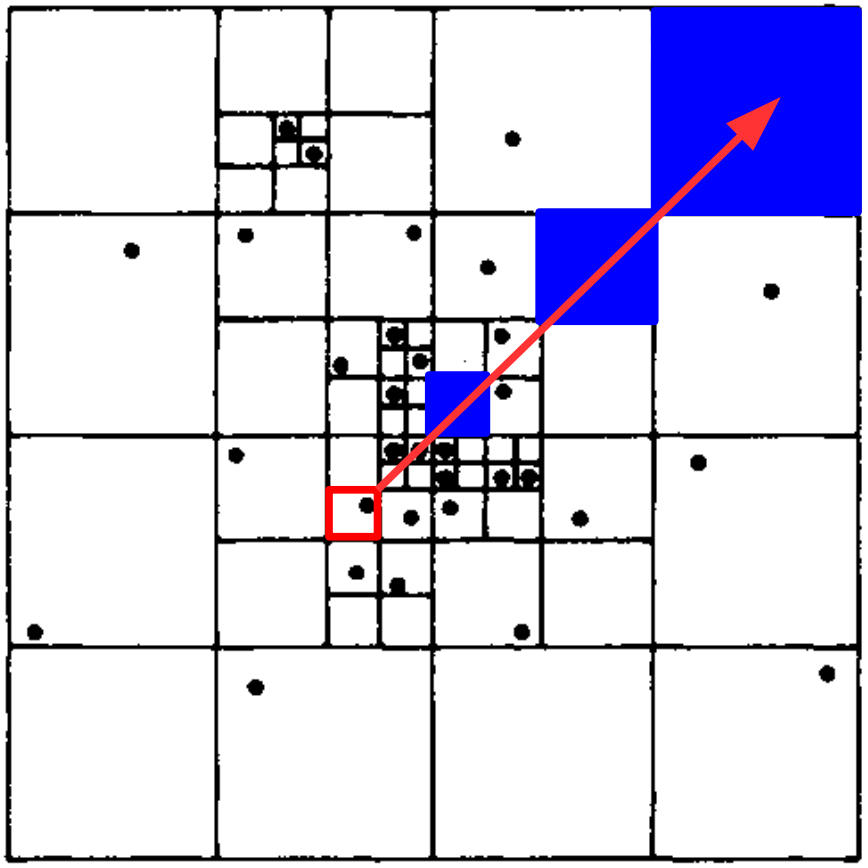
```
def find_accel(self, body, cell):
    """Finds the gravitational effect on 'body' of whatever is inside 'cell'"""
    if (cell.children is None) or (cell.diameter / norm(body.pos - cell.COM) < self.theta):
        return self.newton_accel(body, cell.COM, cell.mass)
    else:
        return sum([self.find_accel(body, child) for child in cell.children])
```

Step Example



- Red squares are added just as N^2 algorithm would
- Green square uses COM of three bodies since diameter is smaller than distance as shown
- Further squares lump more space together into groups of similar subtended angle

O(log(N)) Boxes are Counted



- Each “ring” of 12 boxes triples the area covered
- Thus N operations covers $C*3^{N/12}$ units of area
- So necessary operations go as $\log_3(N/12)\sim\log(N)$ with area
- Similar effect in 3D

Parallelization Notes

- The tree-building half of the algorithm is difficult to efficiently parallelize
  - The tree cannot easily be accessed in parallel
  - The first N levels of the tree can be precomputed, and the construction of the rest of the tree passed to 8^N threads
    - However, resulting cells can be very unbalanced
- The rest of the algorithm, however, just needs to read the tree and update the locations, which are independent and thus easy to parallelize

Parallel Version

- Construct tree:
  - Split list of N particles among available processors, and compute lists of what resides in the first 8^m sub-trees
  - Dynamically load-balance the computation of each of the 8^m sub-trees across available threads
  - Merge sub-trees ($m*8^m$ operations in 1 thread)
- Compute forces:
  - Trivially parallel if each thread has full tree access
    - Tree takes O(N log(N)) memory! May have to be split
  - Otherwise request elements of others' subtrees as necessary
- Update velocities/positions
  - Trivially parallel

Links

- Original Paper:
  - Barnes & Hut, “A hierarchical O(N log N) force-calculation algorithm,”
  - Nature 324, 446 - 449 (04 December 1986); doi:10.1038/324446a0
  - http://www.nature.com/nature/journal/v324/n6096/abs/324446a0.html
- Parallel examples:
  - https://scala-blitz.github.io/home/documentation/examples//barneshut.html
  - http://ta.twi.tudelft.nl/PA/onderwijs/week13-14/Nbody.html
- Gravitational interaction speed:
  - https://arxiv.org/abs/gr-qc/9909087
- Notable N-body simulations:
  - http://hipacc.ucsc.edu/Bolshoi/
  - http://wwwmpa.mpa-garching.mpg.de/millennium/
- Björk concert:
  - “Dark Matter,” Bestival 2011
  - https://www.youtube.com/watch?v=dkagu0qWBio