COMP 633 - Parallel Computing

Lecture 9
September 16, 2021

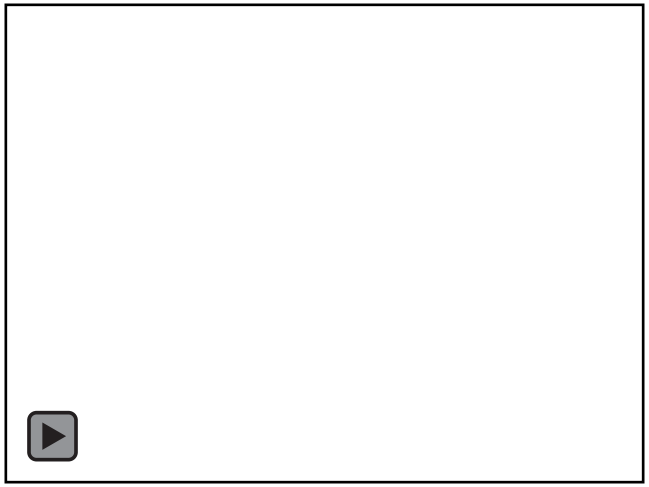
SMM (4)

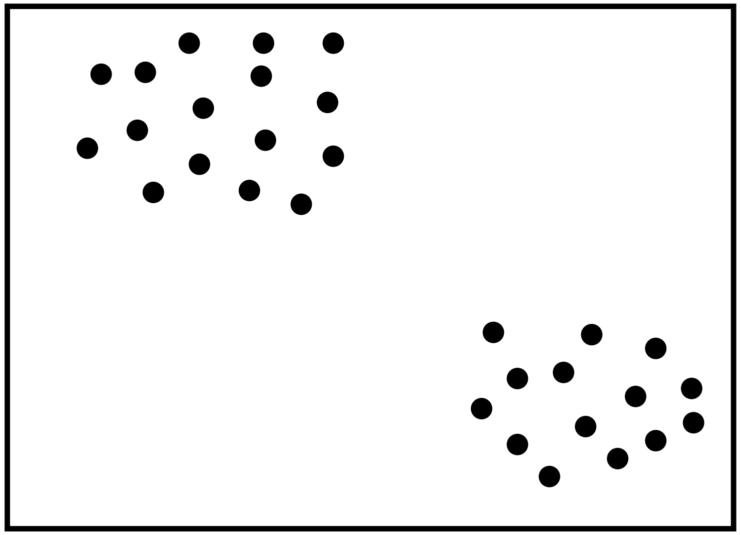
OpenMP Case Study:
The Barnes-Hut N-body Algorithm

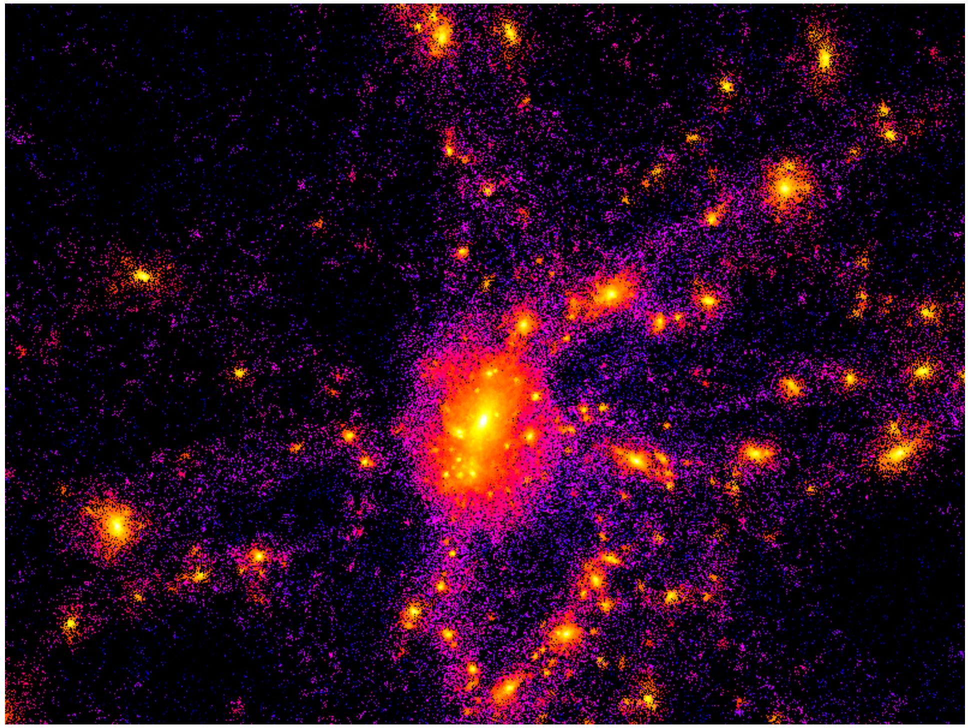
Topics

- Case study: the Barnes-Hut algorithm
  - Study an important method in scientific computing
    - » efficient n-body simulation with long range forces
  - Investigate parallelization and implementation in a shared memory multiprocessor
    - » expression and management of parallelism
    - » memory hierarchy tuning

N-body simulations: self-gravitating systems







The *n*-body simulation problem

- **Simulate the evolution of a system of n bodies over time**
  - Pairwise interaction of bodies
    - » force *f*(*i*,*j*) on body *i* due to body *j*
    - » total force *f*(*i*) on body *i* due to all bodies
    - » acceleration of body *i* via *f* = *ma*
  - Numerical integration of body velocities and positions
    - » timestep Δt
- **Non-negligible long-range forces**
  - for uniformly distributed bodies in 3D, total force due to all bodies at a given distance *r* is constant
    - » cannot ignore contribution of distant bodies
- **Examples**
  - astrophysics (gravity)
  - molecular dynamics (electrostatics)

Ex: Gravitation $r_{ij} = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|$

$$f(i,j) = -G \cdot \frac{m_i \cdot m_j}{r_{ij}^2} \cdot \frac{\mathbf{p}_i - \mathbf{p}_j}{r_{ij}}$$

$$f(i) = \sum_{j \neq i} f(i,j)$$

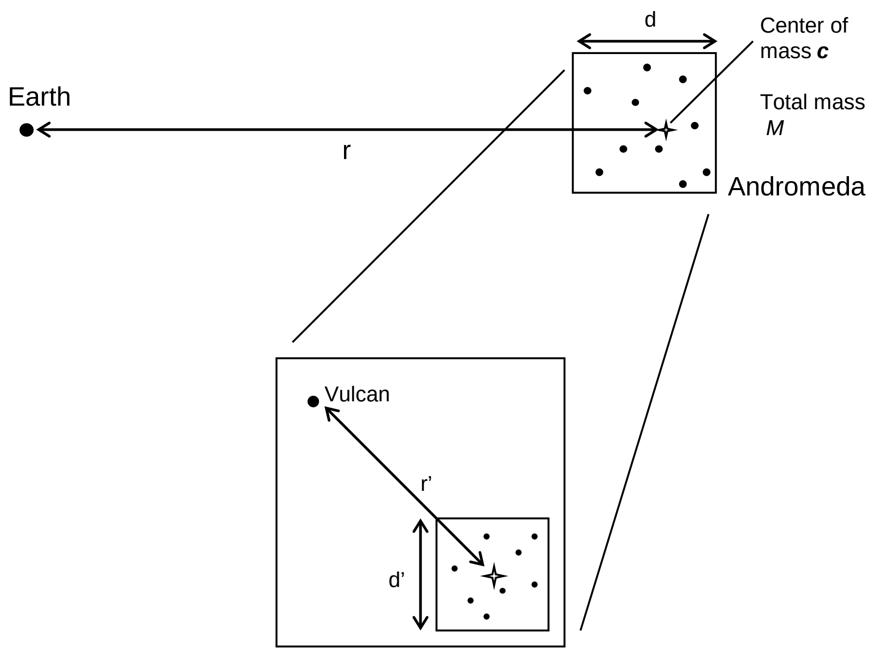
the basic simulation algorithm:

```
while (t < t_Final) do
  forall 1 ≤ i ≤ n do
    ⟨ compute force f(i) on body i ⟩
  end
  ⟨ update velocity and position of all bodies ⟩
  t = t + Δt
end
```

Direct approach:
O(n^2) interactions per time-step

Reducing the number of interactions

Exploit combined effect of “distant” bodies



apply this idea *recursively*:

- determines control-structure
- requires hierarchical decomposition of space

Formally

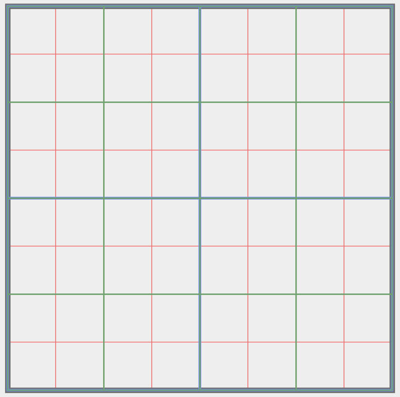
- *Monopole approximation* of the force on the earth due to interaction with all masses in the *Andromeda* galaxy

$$f(b_{\text{earth}}) \approx -G \frac{m_{\text{earth}} M (\mathbf{p}_{\text{earth}} - \mathbf{c})}{r^3}$$

- Monopole approximation saves work if it can be reused with multiple bodies

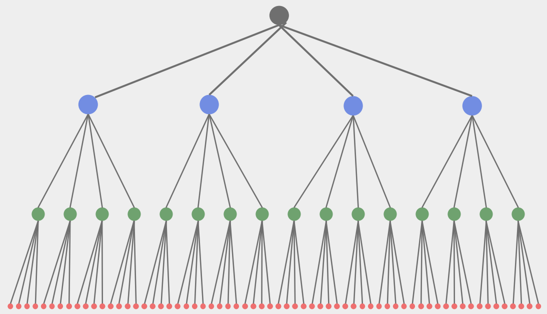
- Accuracy of approximation improves with
  - increasing *r*
  - decreasing *d*
  - order of the approximation
    - » Monopole, dipole, quadropole, …
  - uniformity of body distribution

Hierarchical decomposition of space

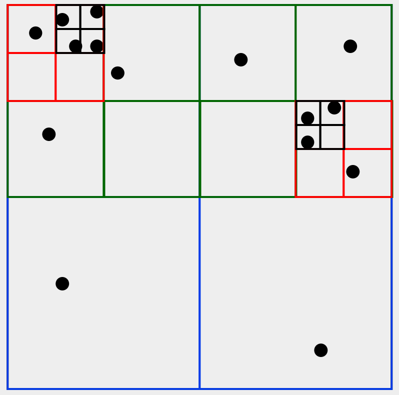


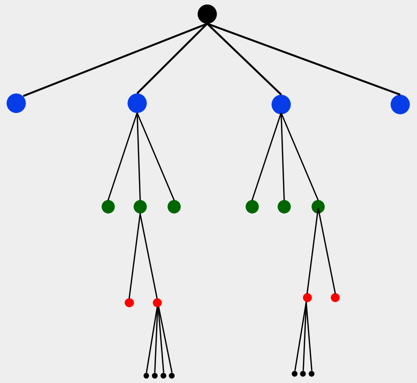
2D

a quadtree

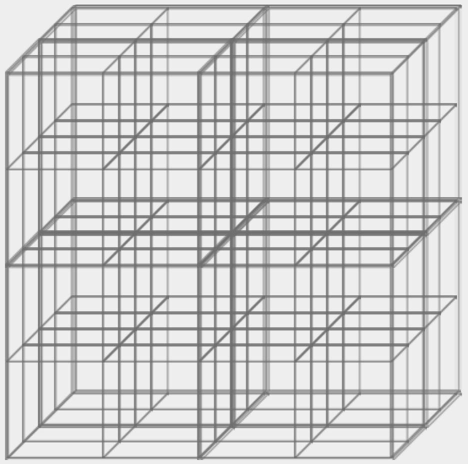


an *adaptive* quadtree





an octree decomposition



3D

The Barnes-Hut algorithm

```
stepSystem():
  // P(i) is coordinates and mass of body i
  T := makeTree(P(1:n))
  forall 1 ≤ i ≤ n do
     f(i) = gravCalc(P(i),T)
  ⟨ update velocities and positions ⟩
```

```
function gravCalc(body p,treenode q)
  if (“q is a leaf”) then
     ⟨return body-body interaction (p,q) ⟩
  else
    if (“p is distant enough from q”) then
       ⟨return body-cell interaction (p,q) ⟩
    else
      forall q’∈ nonemptyChildren(q) do
         accumulate gravCalc(p,q’)
       ⟨return accumulated interaction⟩
    end if
 end if
```

interaction in the case of gravitation:

$$F = G \cdot \frac{m_p \cdot m_q}{r_{pq}^2} \cdot \left[\frac{x_p - x_q}{r_{pq}}, \quad \frac{y_p - y_q}{r_{pq}}, \quad \frac{z_p - z_q}{r_{pq}} \right]$$

$$r_{pq} = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2 + (z_p - z_q)^2}$$

***body-body interaction**: use masses of bodies and distance between them.*

***body-cell interaction**: use mass of body and mass of cell and distance between body and center of mass of cell.*

force is additive; individual contributions can be accumulated.

The Barnes-Hut algorithm - Performance issues

```
stepSystem(P(1:n))
  -- P(1:n) is sequence of bodies
  T := makeTree(P(1:n))
  forall 1 ≤ i ≤ n do
     f(i) := gravCalc(P(i),T)
  ⟨update velocities and positions⟩
```

```
function gravCalc(p,q)
  if (“q is a leaf”) then
    ⟨return body-body interaction⟩
  else
    if (“p is distant enough from q”) then
      ⟨return body-cell interaction⟩
    else
      forall q′∈ nonemptyChildren(q) do
         accumulate gravCalc(p,q′)
      ⟨return accumulated interaction⟩
    end if
 end if
```

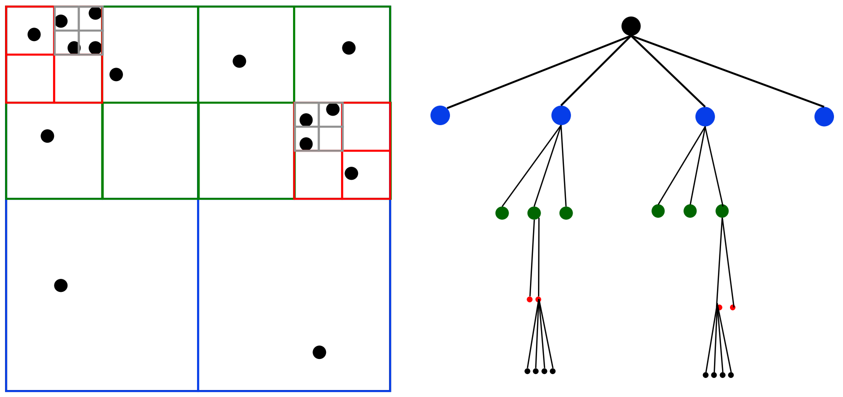
Parallelism

nested parallelism
- over bodies
- over recursively divided cells

load balance
different number of interactions for different bodies

Locality

nearby bodies interact with similar set of nodes in tree

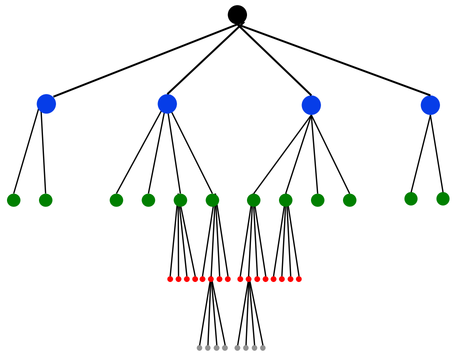


Constructing the tree

- **Small fraction f of the total work**
  - but sequential tree construction can limit overall speedup
    - » Amdahl's law: SP < 1/f
- **Computing monopole approximation for each cell**
  - Post-order traversal of tree
    - » At leaves, monopole coincides with single body
    - » At interior nodes, monopole is weighted sum of all children's monopoles

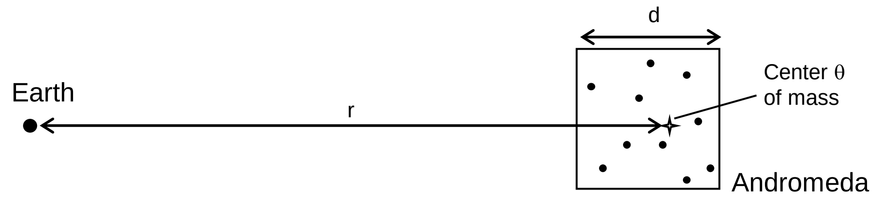
```
function makeTree( P(1:n) )
  for i := 1 to n do
      T := insert(P(i),T)
 〈 compute monopole approximation at each node 〉
```

```
function insert(p,T)
  if empty(T) then
      〈 return p as singleton tree 〉
  else
      〈 determine child S of T in which p belongs 〉
      S' := insert(p,S)
      〈 return T with S replaced by S' 〉
  endif
```



The acceptance criterion

- **when is a cell "distant enough"?**



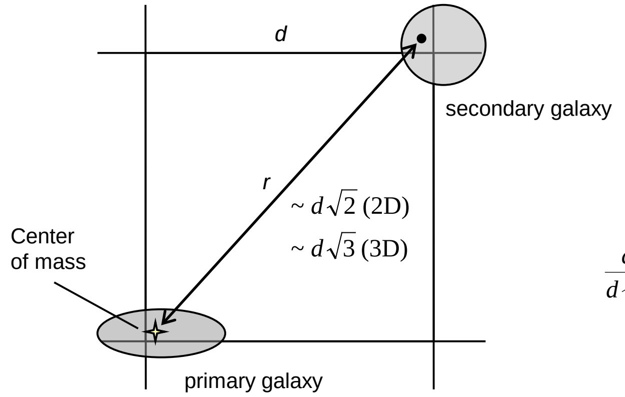
original criterion used by Barnes-Hut:

$$\frac{d}{r} < \theta \equiv r > \frac{d}{\theta}$$

where usually

$$0.7 \leq \theta \leq 1.0$$

- **problem: detonating galaxy anomaly**

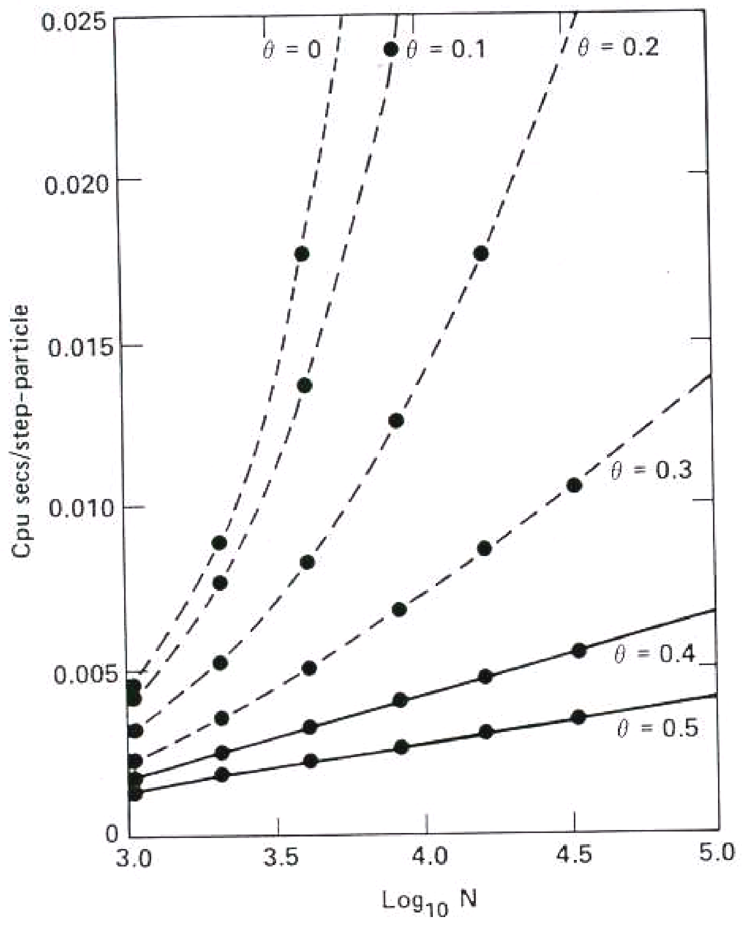


$$\frac{d}{d\sqrt{2}} \approx 0.7 < \theta$$

(one) solution: *add distance between center of mass (cm) and geometric center of cell (c)*

$$r > \frac{d}{\theta} + |cm - c|$$

Effects of acceptance criterion … on runtime



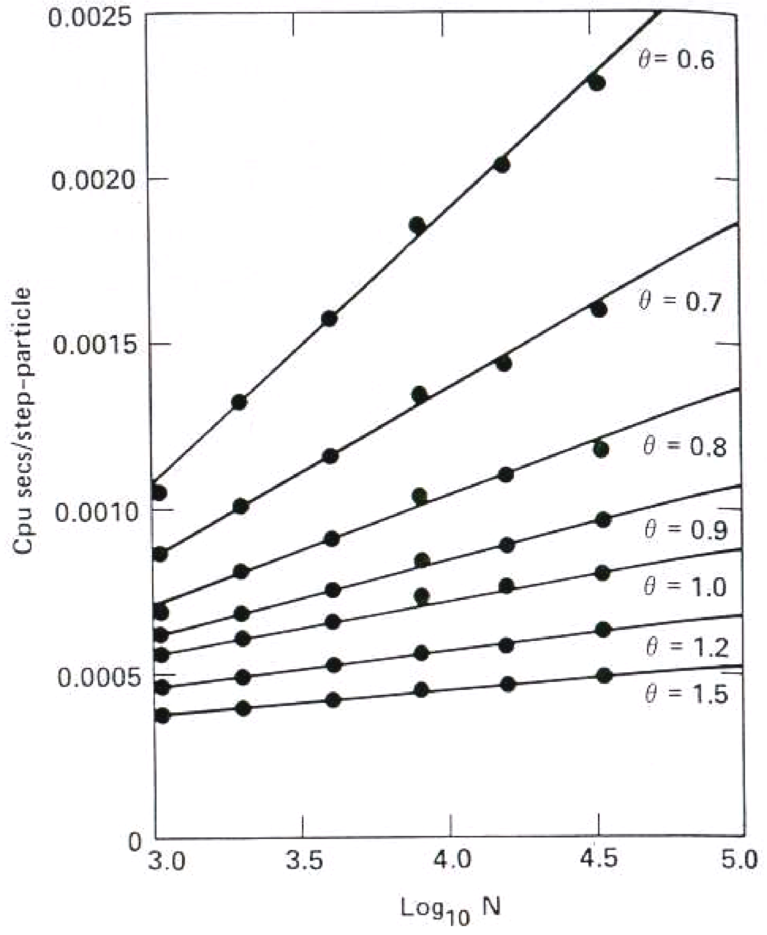


FIG. 3.—Scaling of CRAY X-MP CPU time (CPU seconds per step per particle) for spherical, isotropic Plummer models, as a function of the number of particles, for values of the clumping parameter θ in the range $0 \le \theta \le 1.5$. Only monopole terms have been included in the force computation.

Source: L. Hernquist. *Performance characteristics of tree codes*. Astrophysical Journal Supplement Series, Vol. 64, Pages 715-734, 1987.

Effects of acceptance criterion … on accuracy

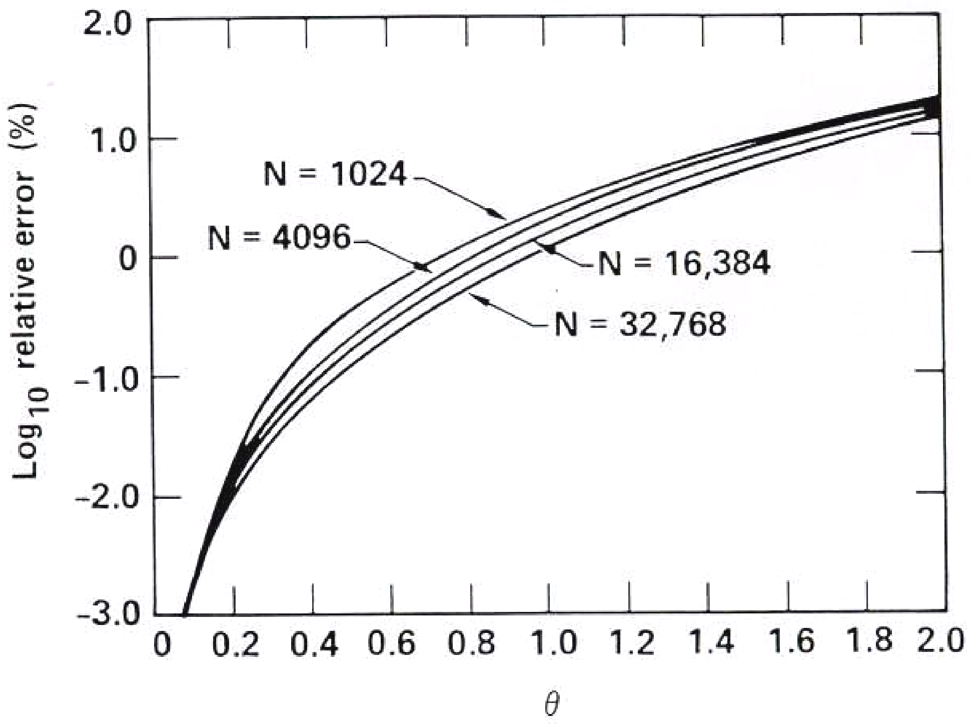
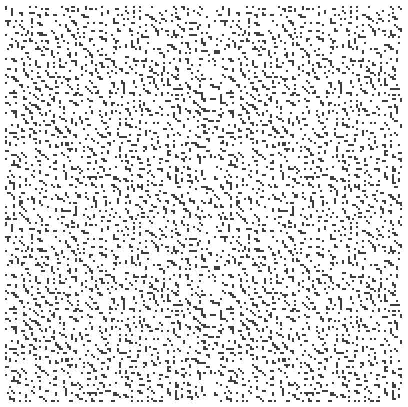


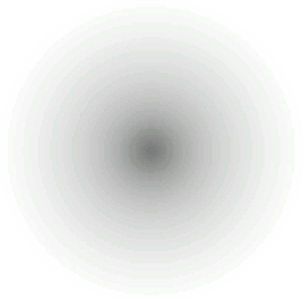
FIG. 6.—Magnitude of the typical error (in percent) in the tree force computation, relative to a direct sum, as a function of θ, for selected values of the particle number N. The calculations have assumed spherical, isotropic Plummer models with softening parameter $\varepsilon = 0$, and only monopole terms have been included in the force computations.

Source: L. Hernquist. *Performance characteristics of tree codes*. Astrophysical Journal Supplement Series, Vol. 64, Pages 715-734, 1987.

1% accuracy sufficient for most astrophysical simulations. Different techniques with better error control necessary for other systems (*fast multipole methods*).

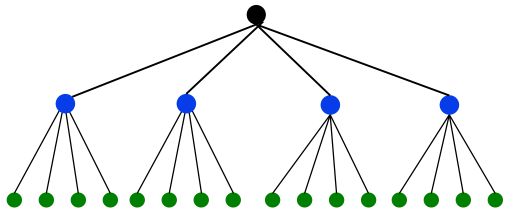
Effect of body distribution … on total work

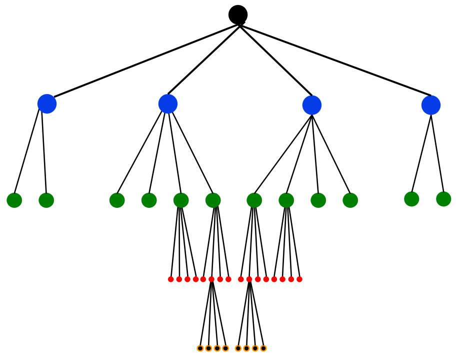




Uniform distribution

Plummer distribution





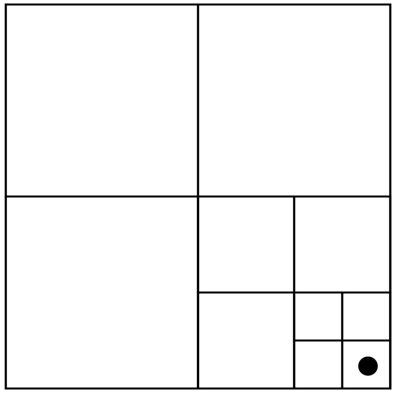
For fixed *n*

- uniform distributions generate high interaction work (shallow trees)
- non-uniform distributions generate higher tree construction and lower interaction work

Complexity of Barnes-Hut

- **Tree building**
  - cost of tree construction depends on particle distribution
    - » cost of body insertion $\propto$ distance to root
    - » for a uniform distribution of *n* particles, sequential construction of the tree is O(*n* log *n*) time
  - In a simulation, tree could be maintained rather than reconstructed each time step

- **Force calculation (uniform distribution of bodies in 2D)**
  - consider computing the force acting on a body in the lower right corner
  - if $\theta = 1.0$ the 3 undivided top-level squares will satisfy the acceptance criterion
  - The remaining square does not satisfy the criterion, hence we descend into the next level
  - each level of the tree incurs a constant amount of work while descending along the path to the lower right corner
  - for a uniform distribution of *n* bodies, the length of the path is O($\log_4 n$)
  - computing the forces on *n* bodies is O(*n* log *n*) work
  - non-uniform distribution more difficult to analyze



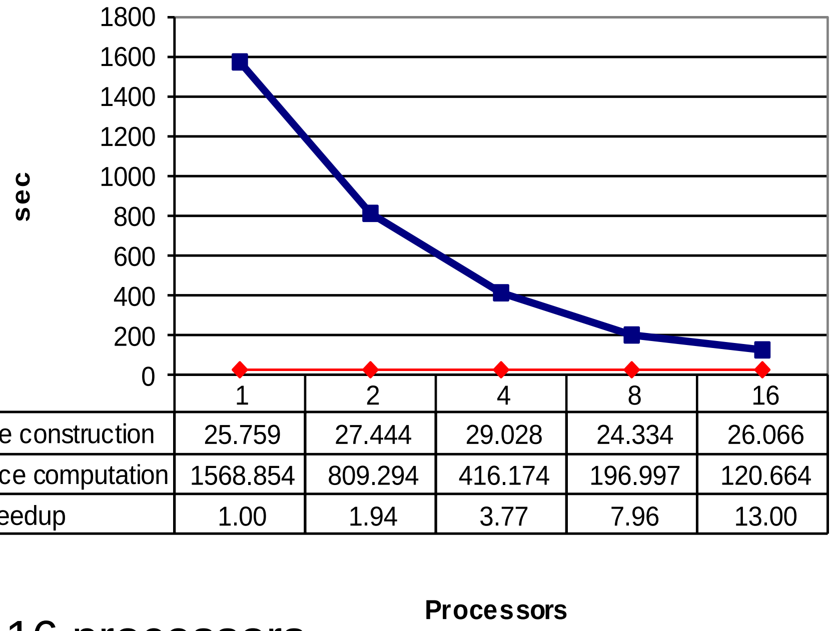
- **Accuracy and complexity are difficult to control**

Implementation issues - parallelization

- **parallelization of the force computation loop:**

```
SUBROUTINE stepSystem()
  CALL makeTree()
  !$OMP PARALLEL DO SCHEDULE(GUIDED,4)
  DO i = 1, n
    CALL gravCalc(i,root)
  END DO
  !$OMP END PARALLEL DO
  !$OMP PARALLEL DO
  ⟨integrate velocities and positions⟩
  !$OMP END PARALLEL DO
END SUBROUTINE stepSystem
```

Results on O2000 (evans) for 1M particles



	1	2	4	8	16
tree construction	25.759	27.444	29.028	24.334	26.066
force computation	1568.854	809.294	416.174	196.997	120.664
speedup	1.00	1.94	3.77	7.96	13.00

- **observations:**
  - force computation scales reasonably up to 16 processors
  - dynamic scheduling important
  - single processor performance not impressive

Implementation issues - tuning of gravCalc (1)

- **performance analysis of gravCalc shows**
  - poor cache reuse (90% L1 and 88% L2)
  - poor use of floating point units
  - poor reuse of subexpressions

  compiler can't generate good code?

- **manual tuning of gravCalc**
  - inline computation of acceptance criterion
  - inline computation of interaction
  - reuse distance vector (body-cell)
  - fuse loops

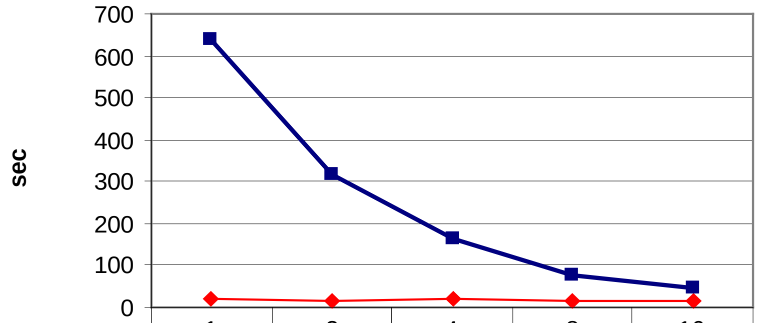
  significant performance improvement!

- **observations:**
  - 2.5 times faster
  - good scaling
  - better use of FPUs and better prediction

  cache reuse (93% L1 and 94% L2) still bad

```
RECURSIVE SUBROUTINE gravCalc(p,q)
  IF (“q is a body”) THEN
    ⟨compute body-body interaction; accumulate⟩
  ELSE
    IF (“p is distant enough from q”) THEN
      ⟨compute body-cell interaction; accumulate⟩
    ELSE
      DO q’ ∈ nonemptyChildren(q)
        CALL gravCalc(p,q’)
      END DO
    END IF
  END IF
END SUBROUTINE gravCalc
```

Results on O2000 (evans) for 1M particles

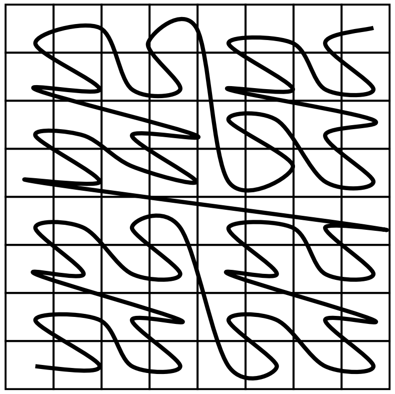


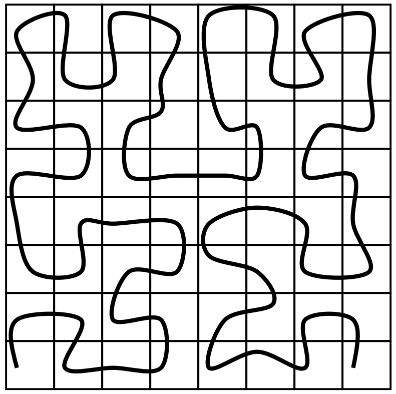
	1	2	4	8	16
🟥 tree construction	19.066	17.878	19.527	15.323	13.686
■ force computation	639.961	315.785	164.764	79.049	44.678
speedup	1.00	2.03	3.88	8.10	14.32

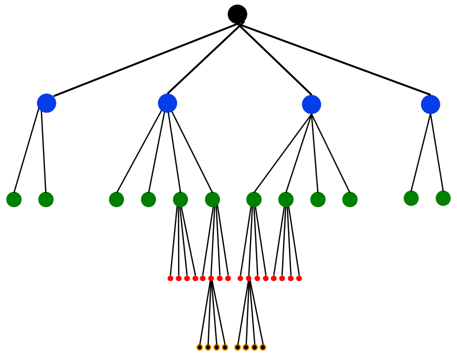
Processors

Implementation issues - tuning of gravCalc (2a)

- **how can we improve cache reuse?**
  - *neighboring bodies in space will most likely interact with the same cells and bodies*!
- **sort bodies according to some spatial order:**
  - precompute spatial order such as Morton order or Peano-Hilbert order
  - or simply order bodies as they are encountered during a depth-first treewalk of T
  - Sorted bodies may also speed up subsequent tree rebuilding


Morton order


Peano-Hilbert order

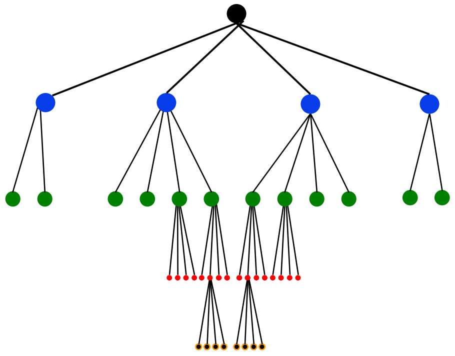

Tree order

Implementation issues - tuning of gravCalc (2b)

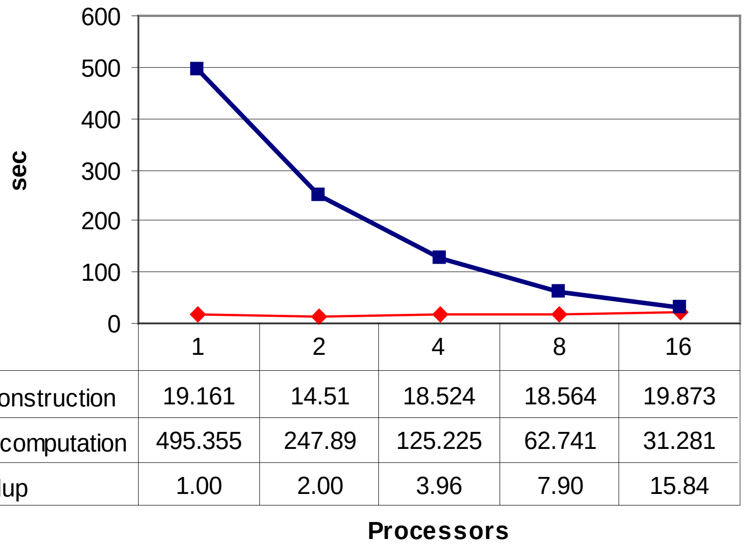
- **observations:**
  - 30-40% increase in performance
  - very good scaling
  - L2 reuse now up at 99.8%
  - L1 still at 93%

```
stepSystem(P(1:n))

   T := makeTree(P(1:n))
   re-order P(1:n) according to T
   forall 1 ≤ i ≤ n do
      f(i) := gravCalc(P(i),T)
   ⟨update velocities and positions⟩
```



Results on O2000 (evans) for 1M particles



	1	2	4	8	16
tree construction	19.161	14.51	18.524	18.564	19.873
force computation	495.355	247.89	125.225	62.741	31.281
speedup	1.00	2.00	3.96	7.90	15.84

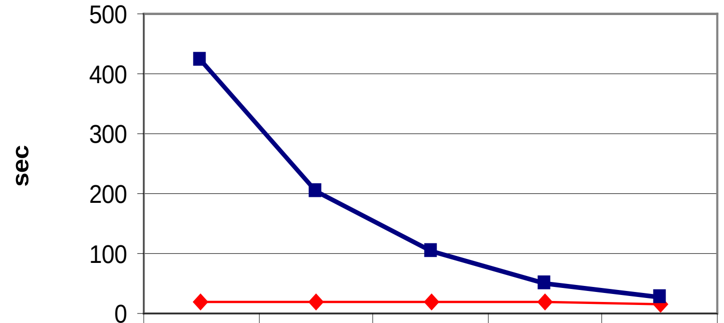
Implementation issues - tuning of gravCalc (3)

How can we improve L1 reuse?

- interact a *group of bodies* with a cell or body!
- walk the tree and compute forces for a *set of neighboring bodies*

```
RECURSIVE SUBROUTINE gravCalc(set P,node q)
  IF ("q is a body") THEN
    DO p ∈ P
      ⟨compute body-body interaction; accumulate⟩
    END DO
  ELSE
    P' = ∅
    DO p ∈ P
      IF ("p is distant enough from q") THEN
        ⟨compute body-cell interaction; accumulate⟩
      ELSE
        P' = P' ∪ {p}
      END IF
    END DO
    IF (P'.NE. ∅) THEN
      DO q' ∈ nonemptyChildren(q)
        CALL gravCalc(P',q')
      END DO
    END IF
  END IF
END SUBROUTINE gravCalc
```

Results on O2000 (evans) for 1M particles



Processors	1	2	4	8	16
🟥 tree construction	20.041	19.471	19.824	18.605	13.716
🟦 force computation	421.391	205.309	104.438	51.828	25.805
speedup	1.00	2.05	4.03	8.13	16.33

Processors

observations:

- 20-40% increase in performance
- L1 reuse now at 99.7% (32 bodies per group)
- L2 down slightly at 96%
- ordered particles essential

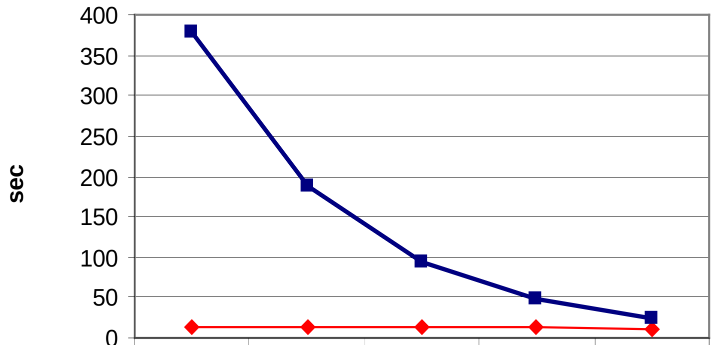
Implementation issues - tuning of gravCalc (4)

Another technique to improve L1 reuse

- allow leaf-cells to contain *more than 1 body*
- compute the body-body interactions in a doubly nested loop.

```
RECURSIVE SUBROUTINE gravCalc(set P, node q)
  P′ = ∅
  DO p ∈ P
    IF (“p is distant enough from q”) THEN
      ⟨compute body-cell interaction; accumulate⟩
    ELSE
      IF (“q is a leaf”) THEN
        DO p ∈ P, q′ ∈ q
          ⟨compute body-body interaction; accumulate⟩
        END DO
      ELSE
        P′ = P′ ∪ {p}
      END IF
    END IF
  END DO
  IF (P′.NE.∅) THEN
    DO q′ ∈ nonemptyChildren(q)
      CALL gravCalc(P′,q′)
    END DO
  END IF
END SUBROUTINE gravCalc
```

Results on O2000 (evans) for 1M particles



	1	2	4	8	16
◆ tree construction	13.179	12.494	13.362	12.682	9.536
■ force computation	378.345	189.231	94.996	47.866	23.809
speedup	1.00	2.00	3.98	7.90	15.89

Processors

observations:

- 10% increase in performance

this algorithm will perform strictly more work than the previous versions! More particles per leaf potentially causes more body-body interactions and fewer body-cell interactions to be computed.

Implementation issues - summary

- **Shared memory model**
  - enables relatively simple parallelization of basic algorithm using OpenMP
  - shared memory model critical in dynamic load balancing
- **Performance tuning**
  - overall these optimizations lead to 4-5 times faster single-processor performance
  - Linear or superlinear parallel speedup to 16 processors
  - optimizing serial performance is essential for obtaining good parallel performance
  - last two optimization are instances of exposing parallelism to improve serial performance
- **Observations**
  - the better the performance of `gravCalc` the more seriously the serial tree-construction affects the overall speedup
    - » when `makeTree` time is included in speedup
      - speedup drops from 13.00 to 10.8 for p = 16 in first version
      - speedup drops from 15.89 to 11.74 for p = 16 on last version
  - parallel tree construction algorithms!