N-Body Simulations – Background

- Suppose the answer at each point depends on data at all the other points
  - Electrostatic, gravitational force
  - Solution of elliptic PDEs
  - Graph partitioning
- Seems to require at least $O(n^2)$ work, communication
- If the dependence on "distant" data can be compressed
  - Because it gets smaller, smoother, simpler…
- Then by compressing data of groups of nearby points, can cut cost (work, communication) at distant points
  - Apply idea recursively: cost drops to O(n log n) or even O(n)
- Examples:
  - Barnes-Hut or Fast Multipole Method (FMM) for electrostatics/gravity/…
  - Multigrid for elliptic PDE
  - …

Fast Multiple Method (FMM)

- "A fast algorithm for particle simulation", L. Greengard and V. Rokhlin, J. Comp. Phys. V. 73, 1987, many later papers
  - Many awards
- Differences from Barnes-Hut
  - FMM computes the *potential* at every point, not just the force
  - FMM uses more information in each box than the CM and TM, so it is both more accurate and more expensive
  - In compensation, FMM accesses a fixed set of boxes at every level, independent of D/r
  - BH uses fixed information (CM and TM) in every box, but # boxes increases with accuracy. FMM uses a fixed # boxes, but the amount of information per box increase with accuracy.
- FMM uses two kinds of expansions
  - Outer expansions represent potential outside node due to particles inside, analogous to (CM,TM)
  - Inner expansions represent potential inside node due to particles outside; *Computing this for every leaf node is the computational goal of FMM*
- First review potential, then return to FMM

Gravitational/Electrostatic Potential

- **FMM will compute a compact expression for potential $\phi(x,y,z)$ which can be evaluated and/or differentiated at any point**
- **In 3D with x,y,z coordinates**
  - **Potential = $\phi(x,y,z) = -1/r = -1/(x^2 + y^2 + z^2)^{1/2}$**
  - **Force = -grad $\phi(x,y,z)$ = - $(d\phi/dx , d\phi/dy , d\phi/dz)$ = $-(x,y,z)/r^3$**
- **In 2D with x,y coordinates**
  - **Potential = $\phi(x,y) = \log r = \log (x^2 + y^2)^{1/2}$**
  - **Force = -grad $\phi(x,y)$ = - $(d\phi/dx , d\phi/dy)$ = $-(x,y)/r^2$**
- **In 2D with z = x+iy coordinates, i = sqrt(-1)**
  - **Potential = $\phi(z) = \log |z| = \text{Real}(\log z)$**

    … because $\log z = \log |z| e^{i\theta} = \log |z| + i\theta$
  - **Drop Real( ) from calculations, for simplicity**
  - **Force = $-(x,y)/r^2 = -z / |z|^2$**

2D Multipole Expansion (Taylor expansion in 1/z) (1/2)

$\phi(z)$ = potential due to z_k, $k=1,\ldots,n-1$

$= \Sigma_k m_k * \log |z - z_k|$

$= \text{Real}(\Sigma_k m_k * \log (z - z_k))$

… since $\log z = \log |z| e^{i\theta} = \log |z| + i\theta$

… drop Real() from now on

$= \Sigma_k m_k * [\log(z) + \log (1 - z_k/z)]$

… how logarithms work

$= M * \log(z) + \Sigma_k m_k * \log (1 - z_k/z)$

… where $M = \Sigma_k m_k$

$= M * \log(z) - \Sigma_k m_k * \Sigma_{s \geq 1} (z_k/z)^s/s$

… Taylor expansion converges if $|z_k/z| < 1$

$= M * \log(z) - \Sigma_{s \geq 1} z^{-s} \Sigma_k m_k z_k^s/s$

… swap order of summation

$= M * \log(z) - \Sigma_{s \geq 1} z^{-s} \alpha_s$

… where $\alpha_s = \Sigma_k m_k z_k^s/s$ … called Multipole Expansion

2D Multipole Expansion (Taylor expansion in 1/z) (2/2)

$\phi(z)$ = potential due to $z_{k,\ k=1,\ldots,n-1}$

$= \Sigma_k\ m_k * \log|z - z_k|$

$= \text{Real}(\ \Sigma_k\ m_k * \log(z - z_k)\ )$

… drop Real() from now on

$= M * \log(z) - \Sigma_{s\geq 1}\ z^{-s}\ \alpha_s$ … Taylor Expansion in 1/z

… where $M = \Sigma_k\ m_k$ = Total Mass and

… $\alpha_s = \Sigma_k\ m_k\ z_k^{\ s}\ /s$

… This is called a Multipole Expansion in z

$= M * \log(z) - \Sigma_{r\geq s\geq 1}\ z^{-s}\ \alpha_s$ + error(r)

… r = number of terms in Truncated Multipole Expansion

… and $\text{error}(\ r\ ) = -\Sigma_{r<s} z^{-s}\ \alpha_s$

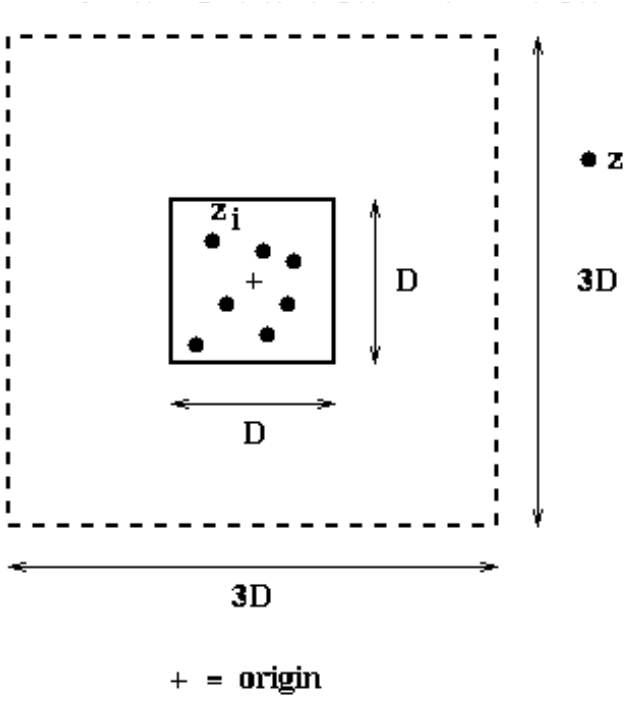
- Note that $\alpha_1 = \Sigma_k\ m_k\ z_k$ = CM*M
  so that M and α_1 terms have same info as Barnes-Hut

- $\text{error}(\ r\ ) = O(\ \{\max_k |z_k| / |z|\}^{r+1}\ )$

Error in Truncated 2D Multipole Expansion

- error(r) = $O(\{\max_k |z_k| / |z|\}^{r+1})$
- Suppose $\max_k |z_k| / |z| \le c < 1$, so
  error(r) = $O(c^{r+1})$
- Suppose all particles z_k lie inside a D-by-D square centered at origin
- Suppose z is outside a 3D-by-3D square centered at the origin
- c = (D/sqrt(2)) / (1.5*D) ~ .47 < .5
  - each term in expansion adds 1 bit of accuracy
  - 24 terms enough for single precision, 53 terms for double precision
- In 3D, can use spherical harmonics or other expansions

Error outside larger box is O(c^(-r))



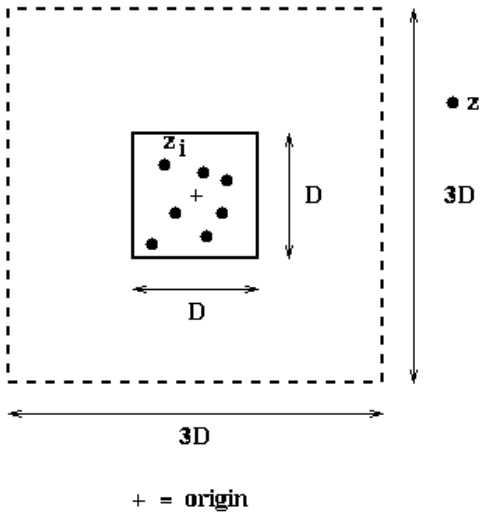
Outer(n) and Outer Expansion

$$\phi(z) \sim M * \log(z - z_n) - \Sigma_{r \geq s \geq 1} (z-z_n)^{-s} \alpha_s$$

- Outer(n) = $(M, \alpha_1, \alpha_2, \ldots, \alpha_r, z_n)$
  - Stores data for evaluating potential $\phi(z)$ outside node n due to particles inside n
  - z_n = center of node n
  - Cost of evaluating $\phi(z)$ is O(r), independent of the number of particles inside n
  - Cost grows linearly with desired number of bits of precision ~r
- Will be computed for each node in QuadTree
- Analogous to (TM,CM) in Barnes-Hut
  - M and α_1 same information as Barnes-Hut

Inner(n) and Inner Expansion

- Outer(n) used to evaluate potential outside node n due to particles inside n
- Inner(n) will be used to evaluate potential inside node n due to particles outside n
  - $\sum_{0 \le s \le r} \beta_s * (z - z_n)^s$
  - z_n = center of node n, a D-by-D box
  - Inner(n) = (β_0 , β_1 , … , β_r , z_n)
  - Particles outside n must lie outside 3D-by-3D box centered at z_n



Top Level Description of FMM

(1) Build the QuadTree

➡ ***(2) Call Build_Outer(root), to compute outer expansions of each node n in the QuadTree***

- … Traverse QuadTree from bottom to top,
- … combining outer expansions of children
- … to get out outer expansion of parent

(3) Call Build_ Inner(root), to compute inner expansions of each node n in the QuadTree

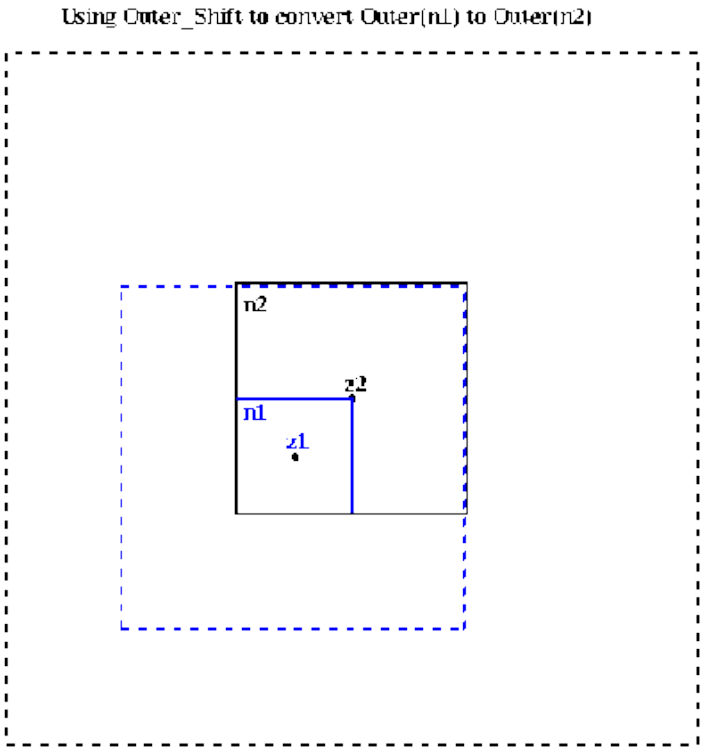
- … Traverse QuadTree from top to bottom,
- … converting outer to inner expansions
- … and combining them

(4) For each leaf node n, add contributions of nearest particles directly into Inner(n)

- … final Inner(n) is desired output: expansion for potential at each point due to all particles

Step 2 of FMM: Outer_shift: converting Outer(n_1) to Outer(n_2) (1/3)

- For step 2 of FMM (as in step 2 of BH) we want to compute Outer(n) cheaply from Outer(c) for all children c of n
- How to combine outer expansions around different points?
  - $\phi_k(z) \sim M_k * \log(z-z_k) - \Sigma_{r \geq s \geq 1} (z-z_k)^{-s} \alpha_{sk}$ expands around z_k , k=1,2
  - First step: make them expansions around same point
- n_1 is a child (subsquare) of n_2
- z_k = center(n_k) for k=1,2
- Outer(n_1) expansion accurate outside blue dashed square, so also accurate outside black dashed square
- So there is an Outer(n_2) expansion with different α_k and center z_2 which represents the same potential as Outer(n_1) outside the black dashed box



Outer_shift: Details (2/3)

- Given

$$\phi_1(z) = M_1 * \log(z-z_1) + \Sigma_{r \geq s \geq 1} (z-z_1)^{-s} \alpha_{s1}$$

- Solve for M_2 and α_{s2} in

$$\phi_1(z) \sim \phi_2(z) = M_2 * \log(z-z_2) + \Sigma_{r \geq s \geq 1} (z-z_2)^{-s} \alpha_{s2}$$

- Get $M_2 = M_1$ and each α_{s2} is a linear combination of the α_{s1}
  - multiply r-vector of α_{s1} values by a fixed r-by-r matrix to get α_{s2}
- $(M_2 , \alpha_{12} , \ldots , \alpha_{r2} , z_2) =$ Outer_shift$(\text{Outer}(n_1) , z_2)$
  $= \text{desired Outer}(n_2)$

Step 2 of FMM: compute Outer(n) for each node n in QuadTree (3/3)

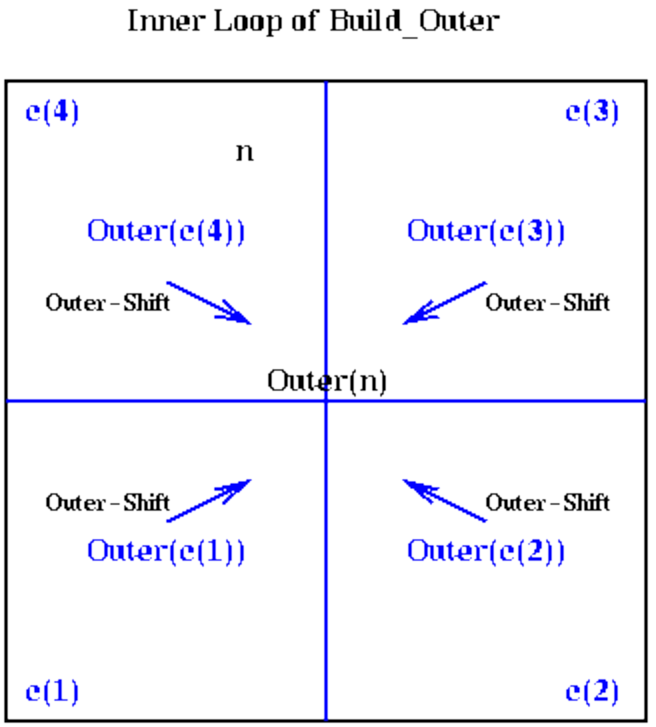
... Compute Outer(n) for each node of the QuadTree
outer = Build_Outer(root)

---

function (M, α_1,...,α_r , z_n) = Build_Outer(n) ... compute outer expansion of node n
    if n if a leaf ... it contains 1 (or a few) particles
        compute and return Outer(n) = (M, α_1,...,α_r , z_n) directly from
            its definition as a sum
    else ... "post order traversal": process parent after all children
        Outer(n) = 0
        for all children c(k) of n ... k = 1,2,3,4
            Outer(c(k)) = Build_Outer(c(k))
            Outer(n) = Outer(n) +
                Outer_shift(Outer(c(k)) , center(n))
                ... just add component by component
        endfor
        return Outer(n)
end if

Cost = O(# nodes in QuadTree) = O(N)

same as for Barnes-Hut



Top Level Description of FMM

(1) Build the QuadTree

(2) Call Build_Outer(root), to compute outer expansions of each node n in the QuadTree

- **… Traverse QuadTree from bottom to top,**
- **… combining outer expansions of children**
- **… to get out outer expansion of parent**

(3) Call Build_ Inner(root), to compute inner expansions of each node n in the QuadTree

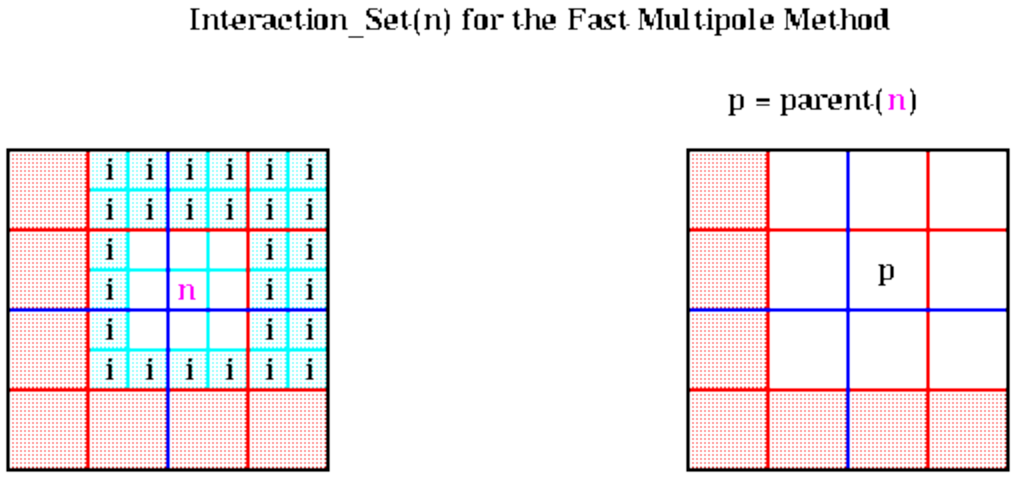
- **… Traverse QuadTree from top to bottom,**
- **… converting outer to inner expansions**
- **… and combining them**

(4) For each leaf node n, add contributions of nearest particles directly into Inner(n)

- **… final Inner(n) is desired output: expansion for potential at each point due to all particles**

Step 3 of FMM: Computing Inner(n) from other expansions

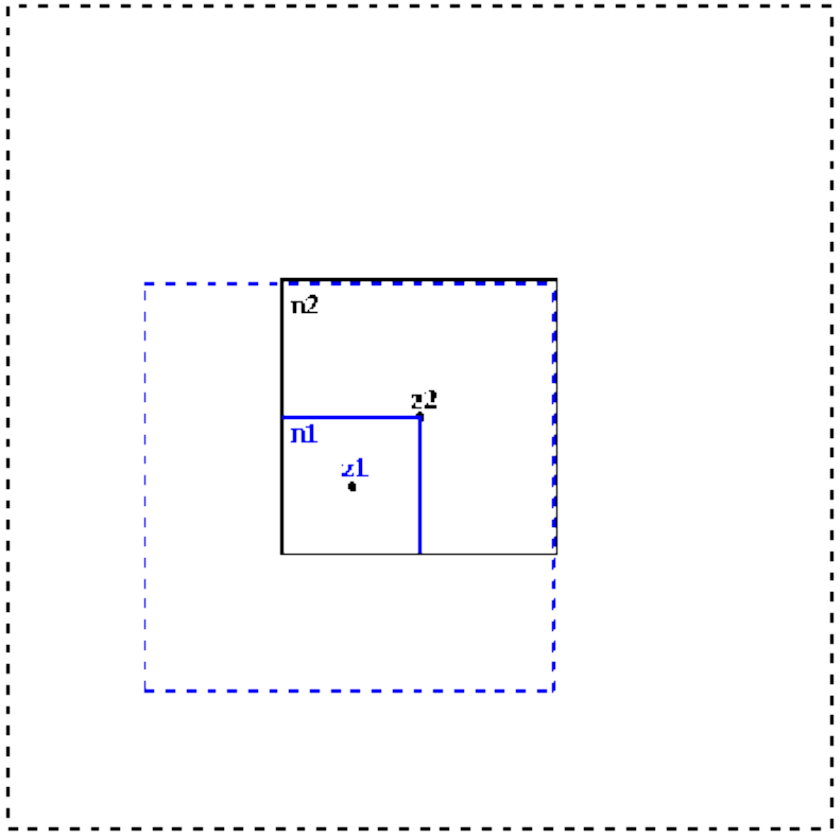
- **Which other expansions?**
  - **As few as necessary to compute the potential accurately**
  - **Inner expansion of parent(n) will account for potential from particles far enough away from parent (red nodes below)**
  - **Outer expansions will account for potential from particles in boxes at same level in Interaction Set (nodes labeled i below)**



Step 3 of FMM: Compute Inner(n) for each n in QuadTree

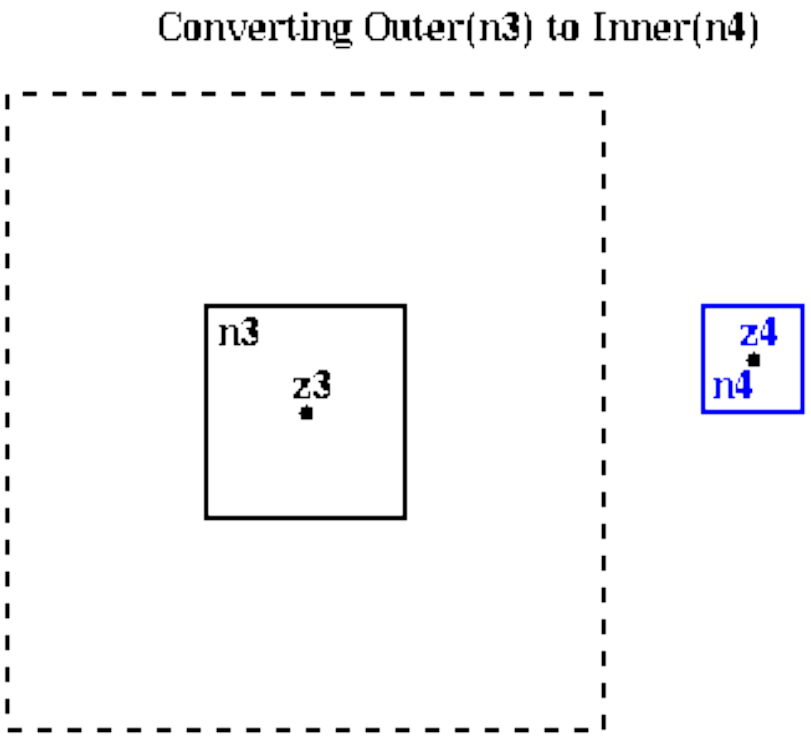
- Need Inner(n_1) = Inner_shift(Inner(n_2), n_1)

Converting Inner(n2) to Inner(n1)



n2 = parent(n1)

- Need Inner(n_4) = Convert(Outer(n_3), n_4)



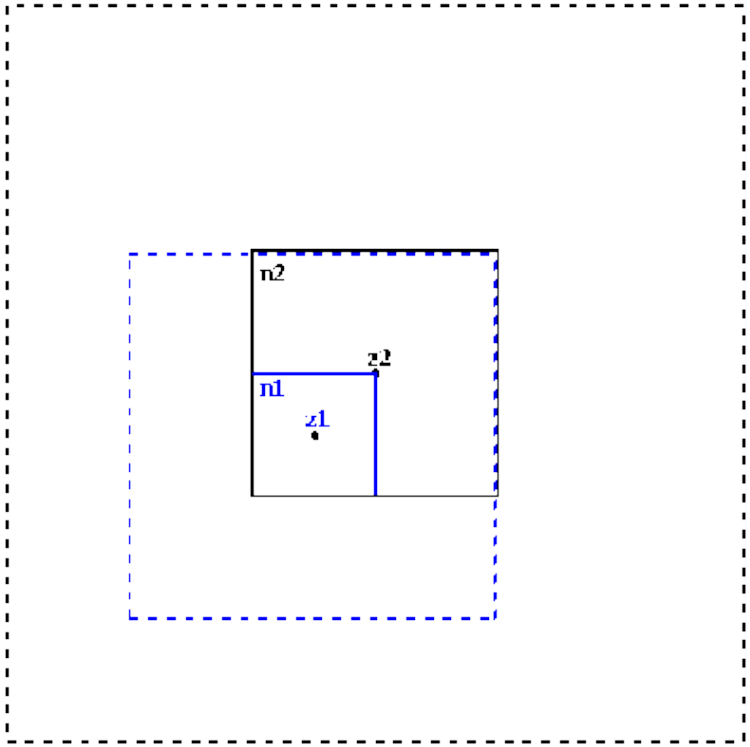
n3 in Interaction_set(n4)

Step 3 of FMM: Inner(n_1) = Inner_shift(Inner(n_2), n_1)

- Inner(n_k) =

  (β_{0k} , β_{1k} , … , β_{rk} , z_k)

Converting Inner(n2) to Inner(n1)



- Inner expansion = $\sum_{0 \le s \le r} \beta_{sk} * (z - z_k)^s$
- Solve $\sum_{0 \le s \le r} \beta_{s1} * (z - z_1)^s = \sum_{0 \le s \le r} \beta_{s2} * (z - z_2)^s$

  for β_{s1} given z_1, β_{s2} , and z_2
  - (r+1) x (r+1) matrix-vector multiply

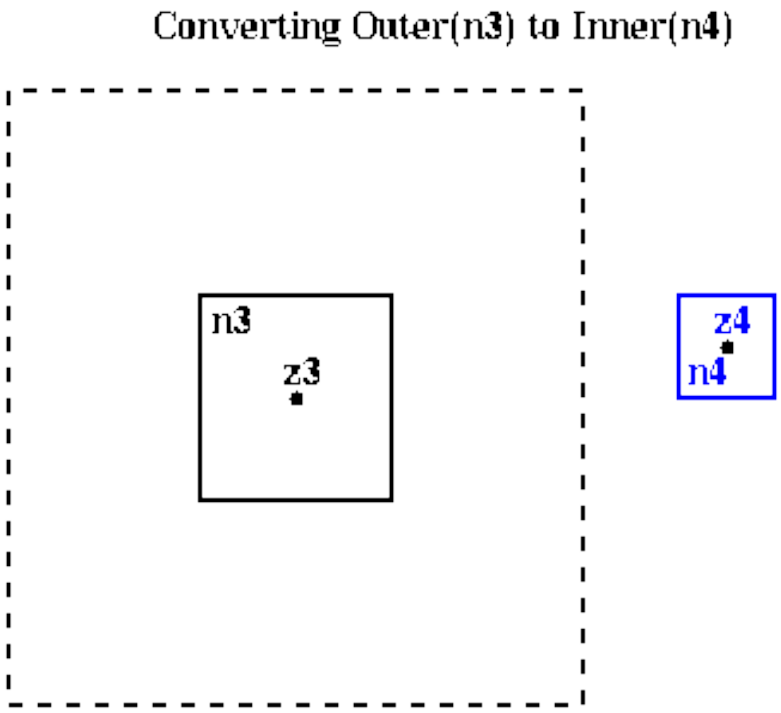
Step 3 of FMM: Inner(n_4) = Convert(Outer(n_3), n_4)

- Inner(n_4) =

  (β_0 , β_1 , … , β_r , z_4)

- Outer(n_3) =

  (M, α_1 , α_2 , … , α_r , z_3)



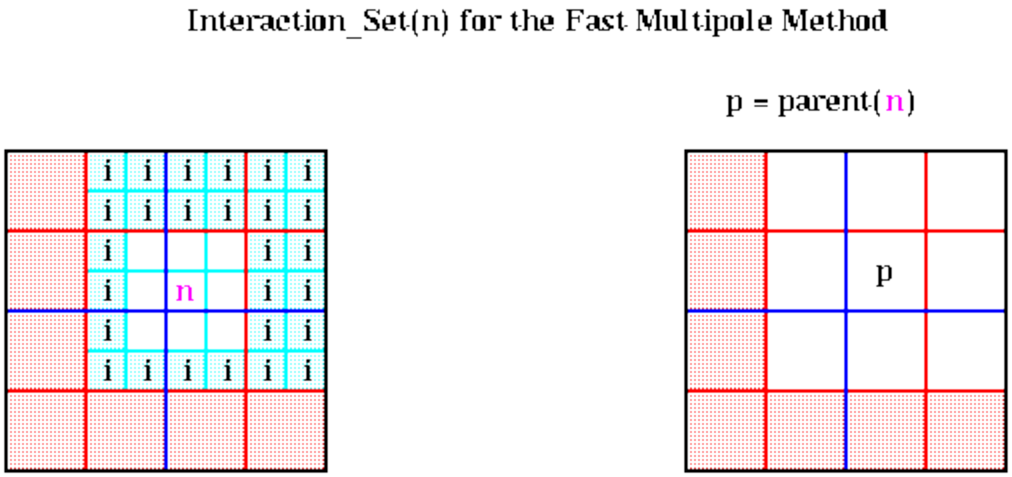
- Solve $\sum_{0 \le s \le r} \beta_s * (z-z_4)^s = M*\log(z-z_3) + \sum_{0 \le s \le r} \alpha_s * (z-z_3)^{-s}$

  for β_s given z_4 , α_e , and z_3

  - (r+1) x (r+1) matrix-vector multiply

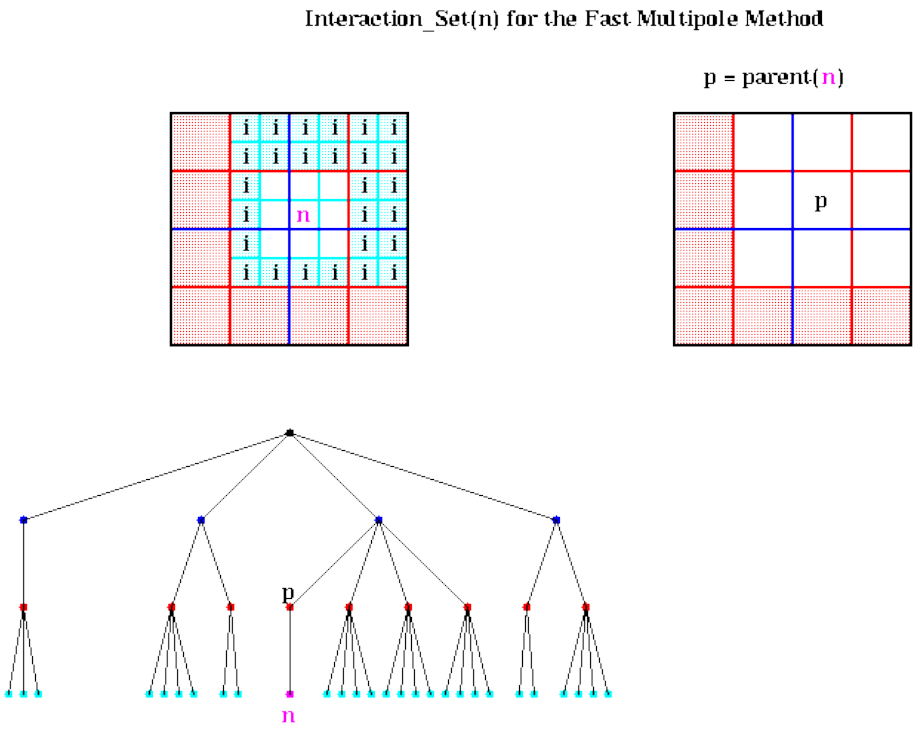
Step 3 of FMM: Computing Inner(n) from other expansions

- We will use Inner_shift and Convert to build each Inner(n) by combing expansions from other nodes
- Which other nodes?
  - As few as necessary to compute the potential accurately
  - Inner_shift(Inner(parent(n)), center(n)) will account for potential from particles far enough away from parent (red nodes below)
  - Convert(Outer(i), center(n)) will account for potential from particles in boxes at same level in Interaction Set (nodes labeled i below)



Step 3 of FMM: Interaction Set

- **Interaction Set = { nodes i that are children of a neighbor of parent(n), such that i is not itself a neighbor of n}**
- **For each i in Interaction Set , Outer(i) is available, so that Convert(Outer(i),center(n)) gives contribution to Inner(n) due to particles in i**
- **Number of i in Interaction Set is at most $6^2 - 3^2 = 27$ in 2D**
- **Number of i in Interaction Set is at most $6^3 - 3^3 = 189$ in 3D**



Step 3 of FMM: Compute Inner(n) for each n in QuadTree

… Compute Inner(n) for each node of the QuadTree
outer = Build_ Inner(root)

---

function ($\beta_1,\ldots,\beta_r$, z_n) = Build_ Inner(n) … compute inner expansion of node n
    p = parent(n) … p=nil if n = root
    Inner(n) = Inner_shift(Inner(p), center(n)) … Inner(n) = 0 if n = root
    for all i in Interaction_Set(n) … Interaction_Set(root) is empty
        Inner(n) = Inner(n) + Convert(Outer(i), center(n))
                … add component by component
    end for
    for all children c of n … complete preorder traversal of QuadTree
        Build_Inner(c)
    end for

Cost = O(# nodes in QuadTree)
= O(N)

Top Level Description of FMM

(1) Build the QuadTree

(2) Call Build_Outer(root), to compute outer expansions of each node n in the QuadTree

- **… Traverse QuadTree from bottom to top,**
- **… combining outer expansions of children**
- **… to get out outer expansion of parent**

(3) Call Build_ Inner(root), to compute inner expansions of each node n in the QuadTree

- **… Traverse QuadTree from top to bottom,**
- **… converting outer to inner expansions**
- **… and combining them**

(4) For each leaf node n, add contributions of nearest particles directly into Inner(n)

- **… if 1 node/leaf, then each particles accessed once,**
- **… so cost = O(N)**
- **… final Inner(n) is desired output: expansion for potential at each point due to all particles**

Parallelizing Hierachical N-Body codes

- **Barnes-Hut, FMM and related algorithm have similar computational structure:**
  - **1) Build the QuadTree**
  - **2) Traverse QuadTree from leaves to root and build outer expansions (just (TM,CM) for Barnes-Hut)**
  - **3) Traverse QuadTree from root to leaves and build any inner expansions**
  - **4) Traverse QuadTree to accumulate forces for each particle**
- **One parallelization scheme will work for them all**
  - **Based on D. Blackston and T. Suel, Supercomputing 97**
    - **UCB PhD Thesis, David Blackston, "Pbody"**
    - **Autotuner for N-body codes**
  - **Assign regions of space to each processor**
  - **Regions may have different shapes, to get load balance**
    - **Each region will have about N/p particles**
  - **Each processor will store part of Quadtree containing all particles (=leaves) in its region, and their ancestors in Quadtree**
    - **Top of tree stored by all processors, lower nodes may also be shared**
  - **Each processor will also store adjoining parts of Quadtree needed to compute forces for particles it owns**
    - **Subset of Quadtree needed by a processor called the Locally Essential Tree (LET)**
  - **Given the LET, all force accumulations (step 4)) are done in parallel, without communication**

Optimizing and Tuning the Fast Multipole Method for Multicore and Accelerator Systems

Georgia Tech
– *Aparna Chandramowlishwaran,* Aashay Shringarpure, Ilya Lashuk; George Biros, Richard Vuduc

Lawrence Berkeley National Laboratory
– Sam Williams, Lenny Oliker

- **Presented at IPDPS 2010**

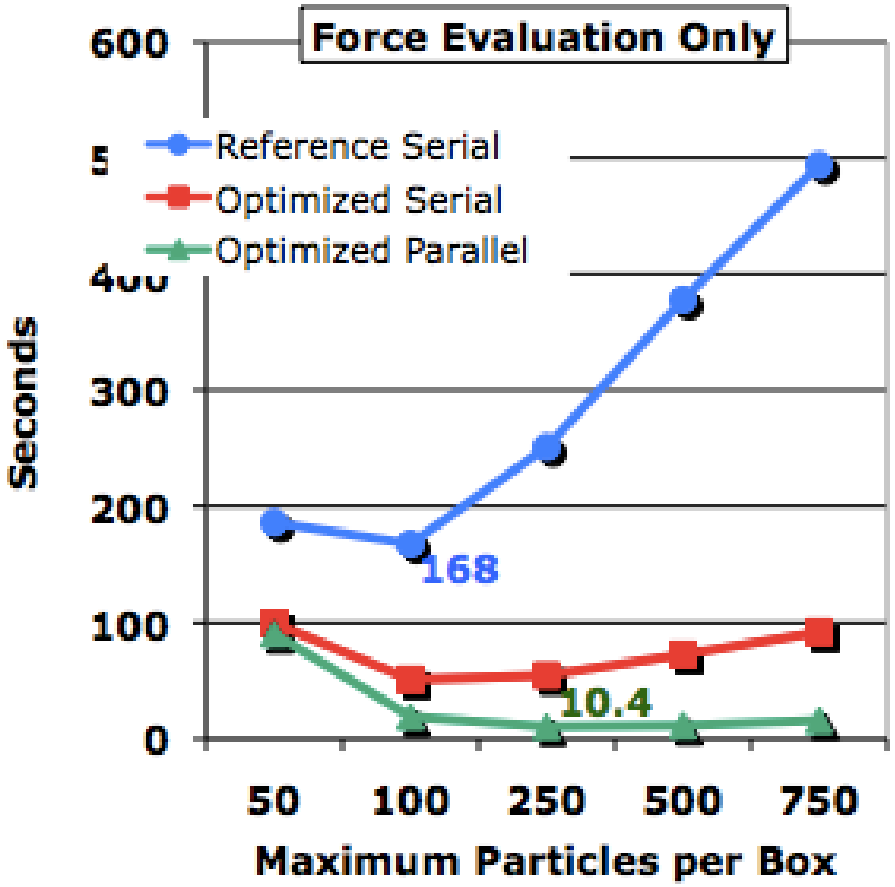
Summary

- First cross-platform single-node multicore study of tuning the fast multipole method (FMM)
  - Explores data structures, SIMD, multithreading, mixed-precision, and tuning
  - Show
    - 25x speedups on Intel Nehalem –
      - 2-sockets x 4-cores/socket x 2-thr/core = 16 threads
    - 9.4x on AMD Barcelona
      - 2-sockets x 4-cores/socket x 1-thr/core = 8 threads
    - 37.6x on Sun Victoria Falls
      - 2-sockets x 8-cores/socket x 8-thr/core = 128 threads

- Surprise? Multicore ~ GPU in performance & energy efficiency for the FMM

Source: Richard Vuduc

Algorithmic Tuning of *q* = Max pts / box - Nehalem



Shape of curve changes as we introduce optimizations.

Source: Richard Vuduc