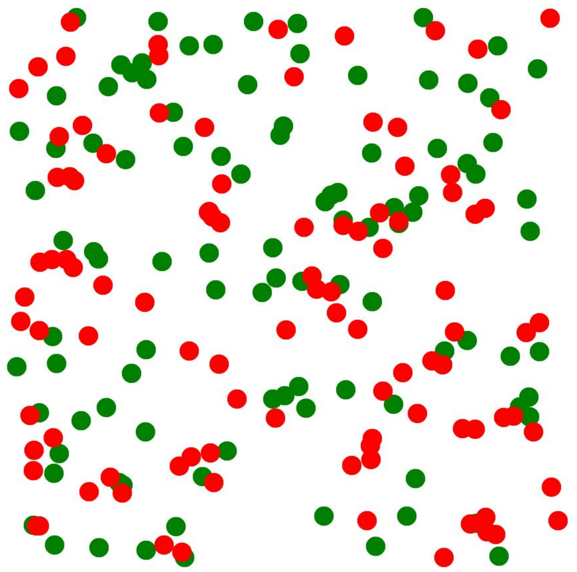
Distributed Fast Multiple Method

Hao Gao

CS598 APK

Dec 13, 2017

Why FMM?



Direct Evaluation – O(MN) – too costly for large problem

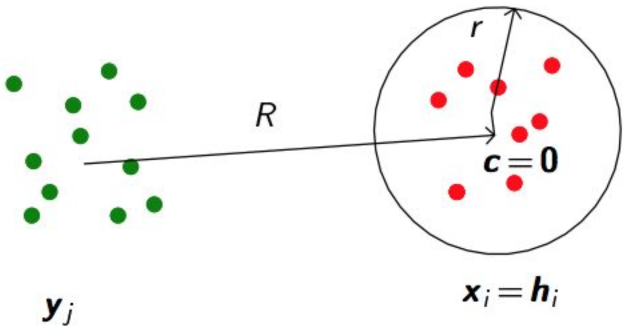
FMM solves this problem in linear time - O(M+N)

In this class, used to evaluate layer potentials

Idea: Local and Multipole Expansion

Local Expansion

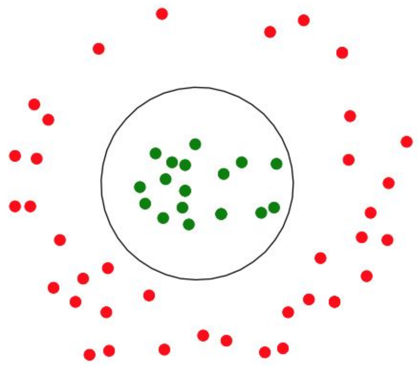
$$\psi(x-y) \approx \sum_{|p|\le k} \frac{D_x^p \psi(x-y)|_{x=c}}{p!}(x-c)^p$$



Error: $\left(\frac{\textit{Furtherst target}}{\textit{Closest source}}\right)^{k+1}$

Multipole Expansion

$$\psi(x-y) \approx \sum_{|p|\le k} \frac{D_y^p \psi(x-y)|_{y=c}}{p!}(y-c)^p$$



Error: $\left(\frac{\textit{Furtherst source}}{\textit{Closest target}}\right)^{k+1}$

Figure Credit: A. Kloeckner

FMM Overview

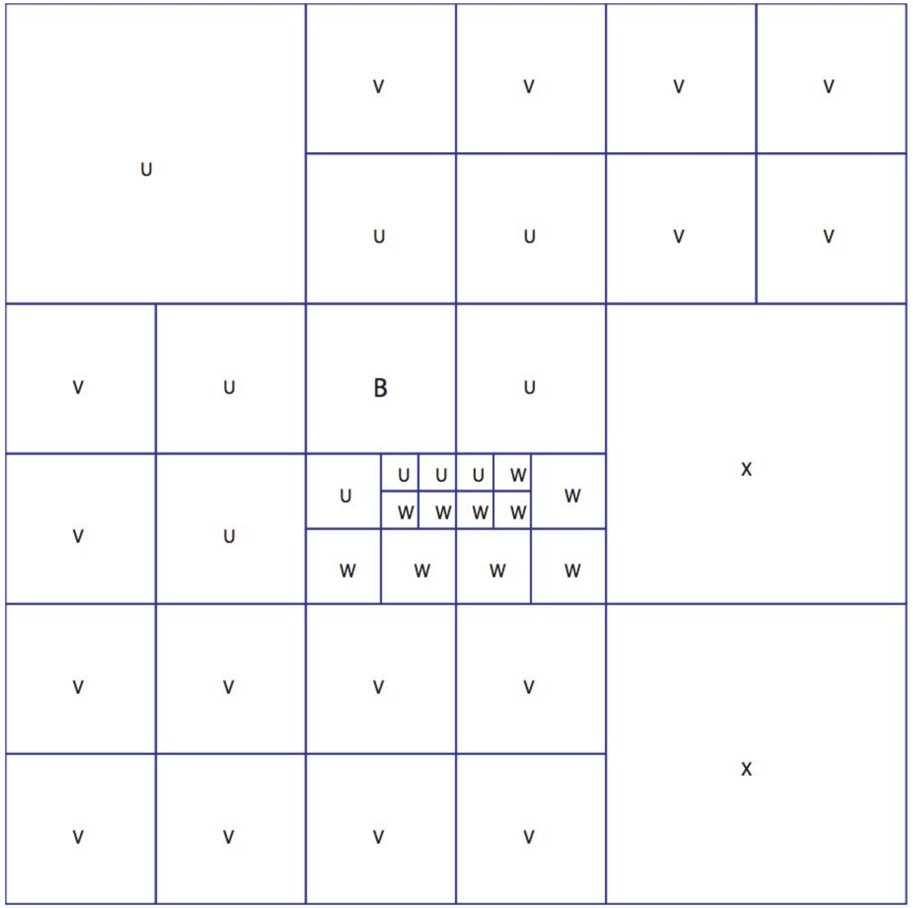
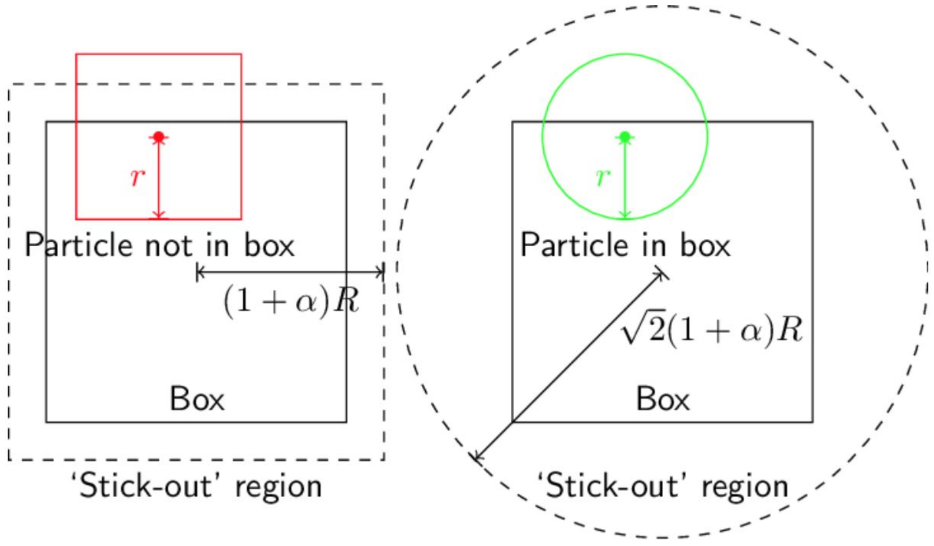


Figure Credit: I. Lashuk, et al.

(1) Build the tree and interaction lists
(2) Calculate multipole densities in the leaf boxes
(3) Upward propagation (M2M)
(4) List 1, U: Direct evaluation
(5) List 2, V: Multipole to local
(6) List 3, W: Multipole to point
(7) List 4, X: Point to local
(8) Downward propagation
(9) Evaluate local expansion at targets

How our FMM is different

Target particles may have scales:



'linf'　　'l2'

- particles on internal nodes
- direct evaluation for some particles on list 3 and 4

Plan of this project

Already have a shared-memory parallel implementation

Time needed to evaluate point potentials of 300,000 sources and 300,000 targets in 2 dimensions, with highest expansion order 3:

Step	Time
Generate Tree	1.45s
Generate Interaction Lists	1.13s
Shared-memory FMM Evaluation (using OpenMP)	13.74s

Distributed FMM Overview

- Build the tree and interaction list on the root process
- Work decomposition: process i assigned "responsible boxes" ($\mathcal{L}_i$)
- Distribute the structure of the whole tree with a subset of particles to each process
- Compute multipole densities in $\mathcal{L}_i$ and upward propagation
- Communicate densities across all processes
- *(Each process has all information needed for FMM evaluation)*
- Evaluate M2L, P2L on $\mathrm{A}(\mathcal{L}_i)$
- Evaluate step (4) – (9) using shared-memory FMM for all targets in $\mathcal{L}_i$

What particles to distribute, and how?

- All sources and targets in $\mathcal{L}_i$
- Sources in List1, List3 near, List4 near of $\mathcal{L}_i$ (Direct evaluation)
- Sources in List 4 of $\mathcal{L}_i$ (P2L)
- Sources in List 4 of all ancestors of $\mathcal{L}_i$ (P2L, downward)

```
count ← 0
for i ← 1 to nparticles do
    if particle[i] ∈ S then
        A[count] ← particle[i]
        count ← count + 1
```

5	x	7	3	6	x	4	x	x	1

1	0	1	1	1	0	1	0	0	1

0	1	1	2	3	4	4	5	5	5

5	7	3	6	4	1

Load Balancing

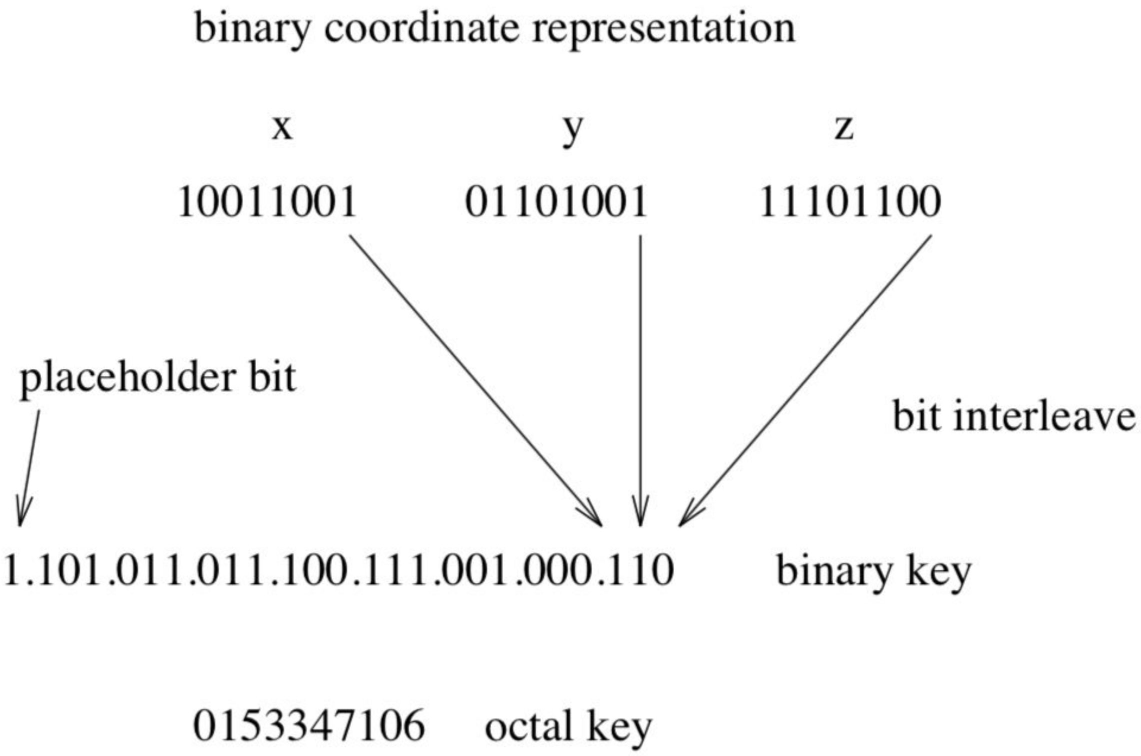
- First try: Divide all boxes evenly
- Second try: Divide all particles evenly
- Current scheme: use DFS (Morton) order, divide the workload evenly

$$\mathcal{W}(x) = \alpha|x| + \beta \sum_{y \in U(x)} |x||y|$$

$\mathcal{W}(x)$:= Workload of x
$|x|$:= #particles in x
$U(x)$:= List1, List3 near, List4 near of x

FMM in 1 thread	51.88s
process 1 of 8	5.32s
process 2 of 8	5.85s
process 3 of 8	5.86s
process 4 of 8	5.97s
process 5 of 8	6.69s
process 6 of 8	6.65s
process 7 of 8	7.47s
process 8 of 8	7.80s

Morton (DFS) ordering



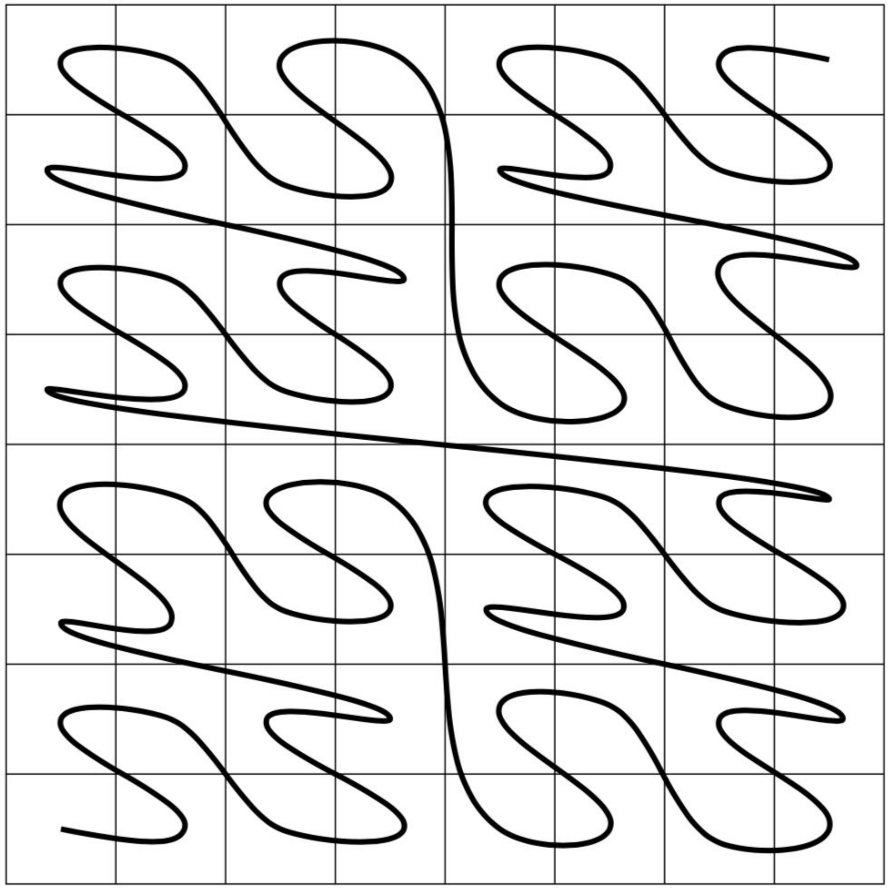


Figure Credit: M. Warren & J. Salmon

Communication in upward propagation

- Can use an MPI_Allreduce, but not efficient
- Process i is a contributor of box β if $\beta \in \mathcal{L}_i \cup \mathrm{A}(\mathcal{L}_i)$
- Process i is a user of box β if $\beta \in V(\mathcal{L}_i) \cup W(\mathcal{L}_i)$
- Box β needs to be sent from process i to process j iff process i is a contributor and process j is a user
- Even better: tree based communication pattern

Future plan

- Reorder the box to save particle scan
- Integrate with layer potential evaluation
- Test scalability on large scale of processors
- Overlap communication and computation

Reference

Lashuk, I., Chandramowlishwaran, A., Langston, H., Nguyen, T. A., Sampath, R., Shringarpure, A., ... & Biros, G. (2012). A massively parallel adaptive fast multipole method on heterogeneous architectures. *Communications of the ACM*, *55*(5), 101-109.

Warren, M. S., & Salmon, J. K. (1993, December). A parallel hashed oct-tree n-body algorithm. In *Proceedings of the 1993 ACM/IEEE conference on Supercomputing* (pp. 12-21). ACM.