Fast Multipole Method

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Quantitative & Computational Biology
University of Southern California

Email: anakano@usc.edu

Objectives: Space-time multiresolution algorithms
> Tree codes: fast multipole method
> Multiple time stepping

Top 10 Algorithms in History

In putting together this issue of *Computing in Science & Engineering*, we knew three things: it would be difficult to list just 10 algorithms; it would be fun to assemble the authors and read their papers; and, whatever we came up with in the end, it would be controversial. We tried to assemble the 10 algorithms with the greatest influence on the development and practice of science and engineering in the 20th century. Following is our list (here, the list is in chronological order; however, the articles appear in no particular order):

- Metropolis Algorithm for Monte Carlo
- Simplex Method for Linear Programming
- Krylov Subspace Iteration Methods
- The Decompositional Approach to Matrix Computations
- The Fortran Optimizing Compiler
- QR Algorithm for Computing Eigenvalues
- Quicksort Algorithm for Sorting
- Fast Fourier Transform
- Integer Relation Detection
- Fast Multipole Method

PHYS 516
CSCI 596
CSCI 653

IEEE CiSE **2(1)**, 22 ('00)

https://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=17639
https://aiichironakano.github.io/cs653/Greengard-FMM-JCP87.pdf

Data Locality in MD

- Spatial locality: Atoms closer to each other interact more tightly
- Temporal locality: Computations performed in consecutive MD time steps are similar

  ↓ Take advantage!

- Efficient simulation algorithms
  > Reduced computational complexity
  > Better utilization of hardware memory hierarchy (intranode)
  > Minimized communication overhead (internode)

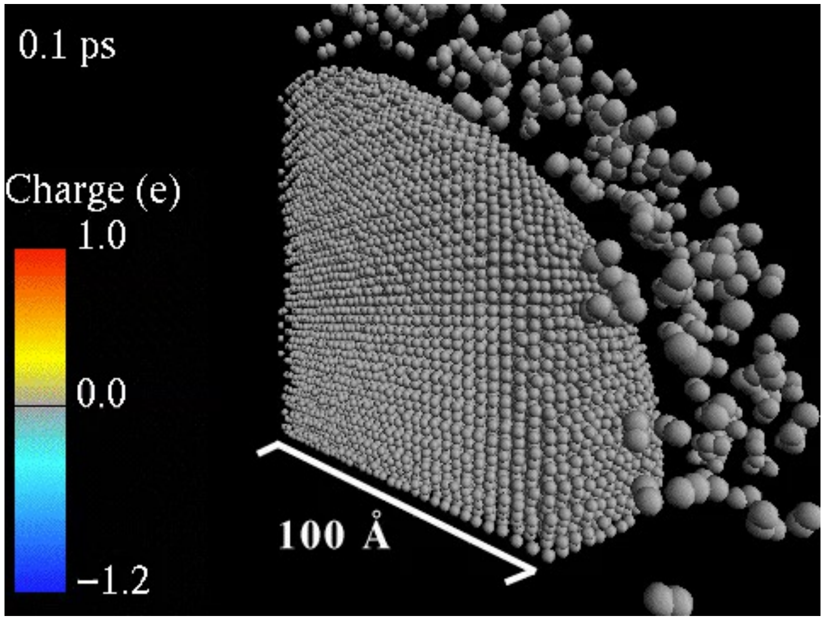
Molecular Dynamics: *N*-Body Problem

- **Newton's equations of motion**

$$m_i \frac{d^2\mathbf{r}_i}{dt^2} = -\frac{\partial E_{\mathrm{MD}}\left(\mathbf{r}^N\right)}{\partial \mathbf{r}_i} \qquad (i = 1,\ldots,N)$$

- **Reliable interatomic potential**

$$E_{\mathrm{MD}} = \sum_{i<j} u_{ij}\left(r_{ij}\right) + \sum_{i,j<k} v_{jik}\left(\mathbf{r}_{ij},\mathbf{r}_{ik}\right)$$



- ***N*-body problem**
  Long-range electrostatic interaction — *O*(*N*²)

$$\text{Evaluate } V_{\mathrm{es}}(\mathbf{x}) = \sum_{i=1}^{N} \frac{q_i}{|\mathbf{x}-\mathbf{x}_i|} \text{ at } \mathbf{x} = \mathbf{x}_j \ (j = 1,\ldots,N)$$

- ***O*(*N*) space-time multiresolution MD (MRMD) algorithm**
  1. **Fast multipole method (FMM)** [Greengard & Rokhlin, '87]
  2. **Symplectic multiple time stepping (MTS)** [Tuckerman *et al.*, '92]

Clustering in the Fast Multipole Method

- **Encapsulate far-field information in terms of multipoles at the source**

$$V(\mathbf{x}) = \sum_{i=1}^{N} \frac{q_i}{|\mathbf{x}-\mathbf{x}_i|}$$

$$= \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \underbrace{\left\{ \sum_{i=1}^{N} q_i r_i^l Y_l^{*m}(\theta_i,\phi_i) \right\}}_{\text{multipole}} \frac{Y_l^m(\theta,\phi)}{r^{l+1}}$$

Spherical harmonics

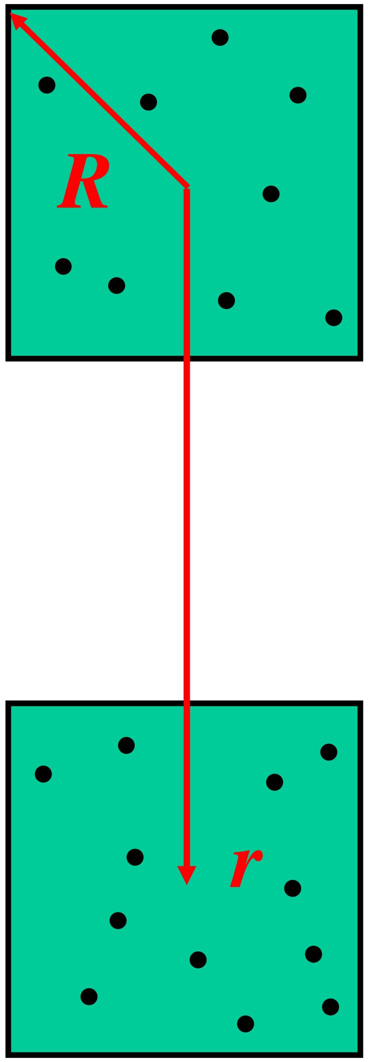
- **Well-defined error bound**

$$\text{Error} \le \frac{\sum_{i=1}^{N} |q_i|}{r-R} \left(\frac{R}{r}\right)^{p+1}$$

p: l-sum truncation

- **Local expansion at the destination**

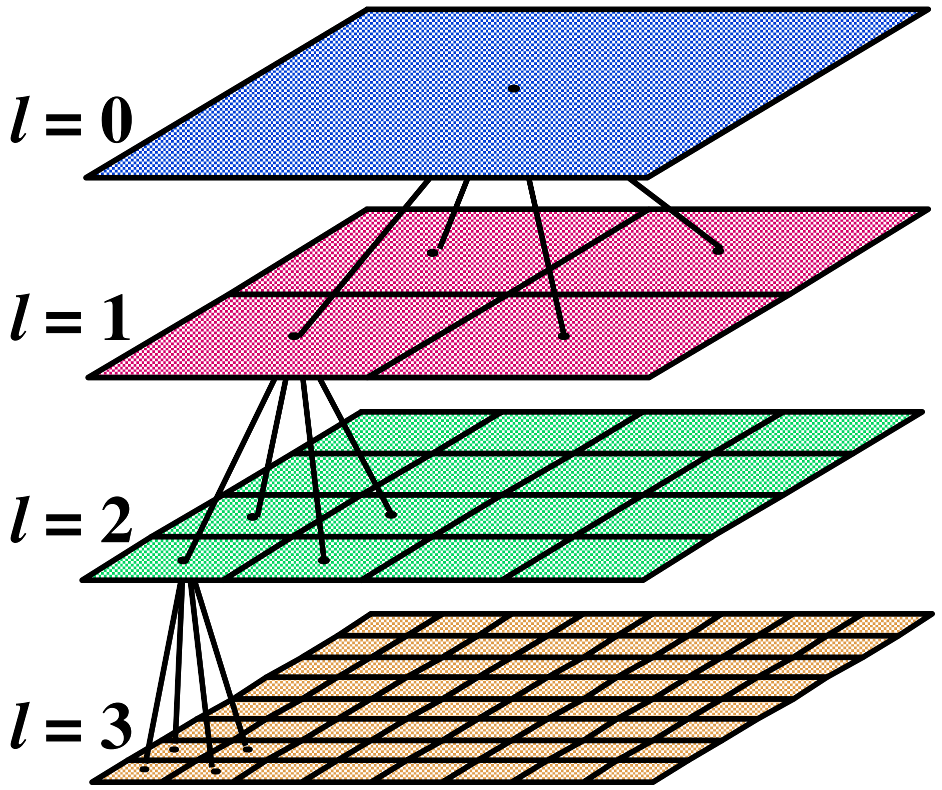
$$V(\mathbf{x}) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \underbrace{\left\{ \sum_{i=1}^{N} \frac{q_i Y_l^m(\theta_i,\phi_i)}{r_i^{l+1}} \right\}}_{\text{local term}} r^l Y_l^{*m}(\theta,\phi)$$



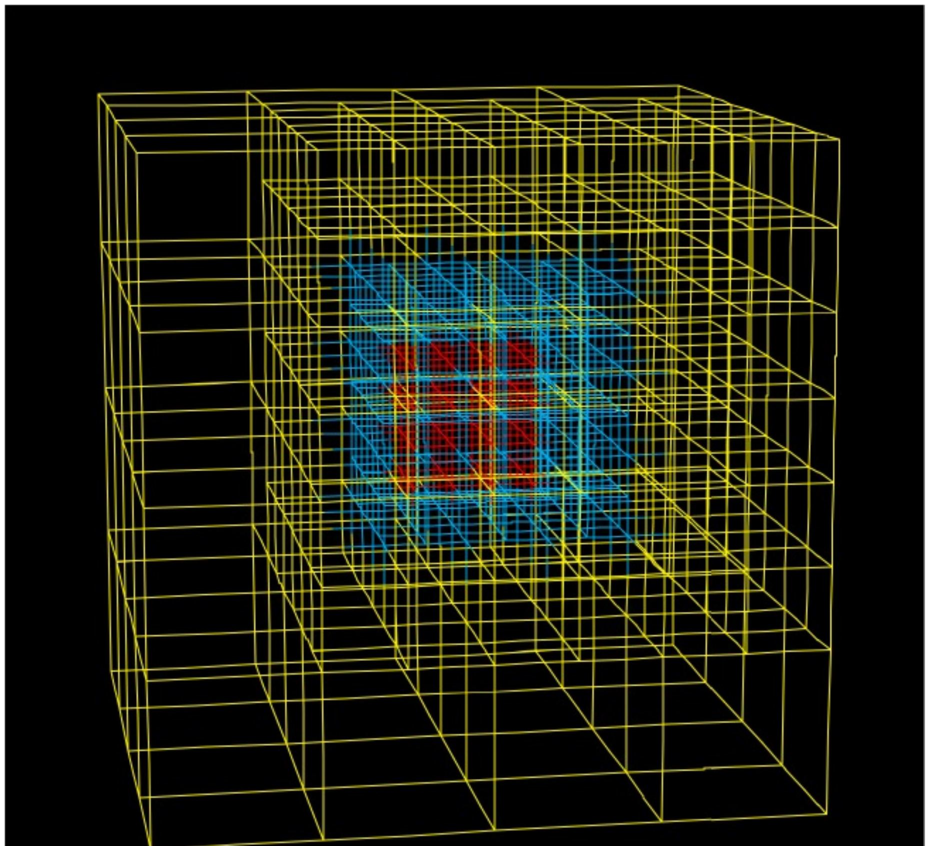
L. Greengard & V. Rokhlin, *J. Comput. Phys.* **73**, 325 ('87)

https://aiichironakano.github.io/cs653/Greengard-FMM-JCP87.pdf

Hierarchical Abstraction



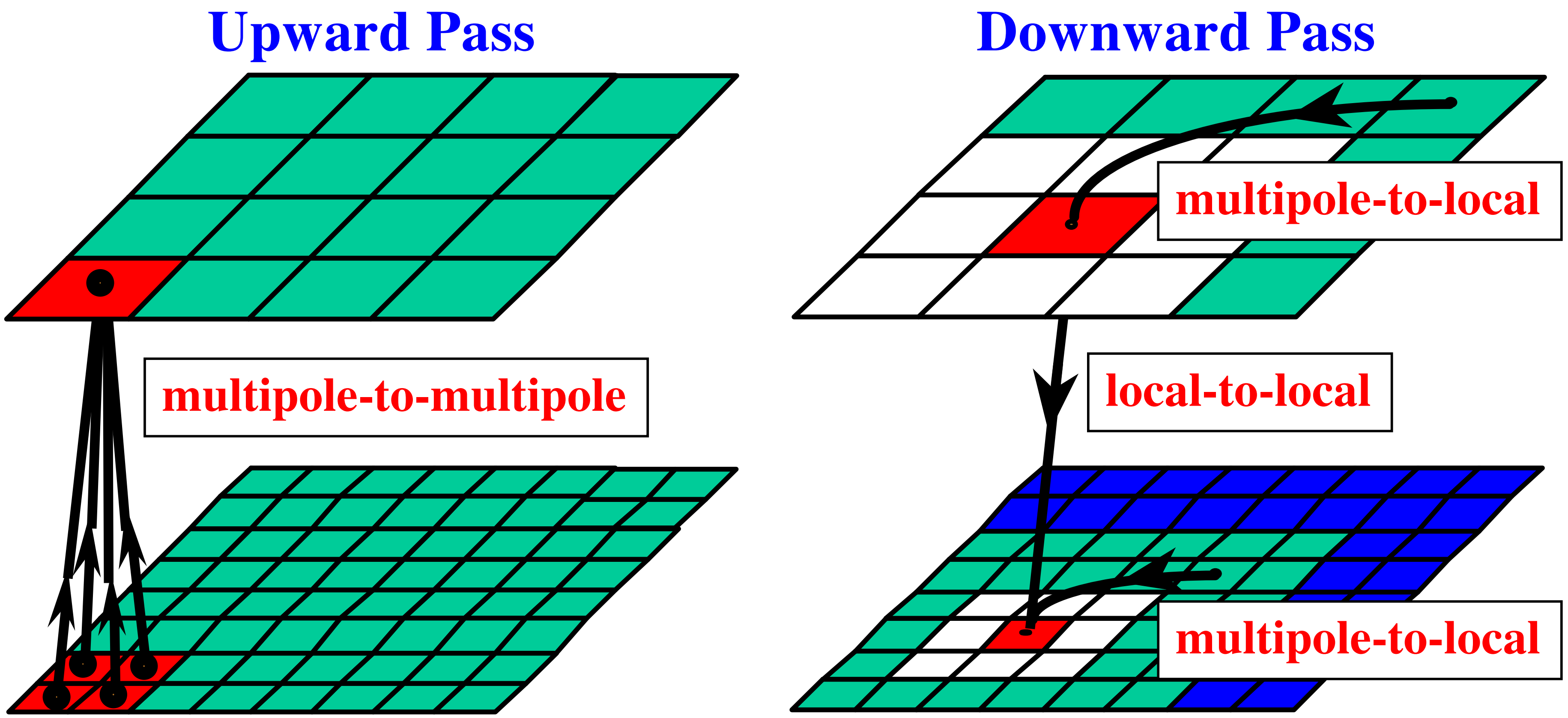
2D example



Use of an octree

- Larger clusters for longer distances to keep the error constant
- Recursively subdivide the simulation box to form an octree in 3D or quadtree in 2D

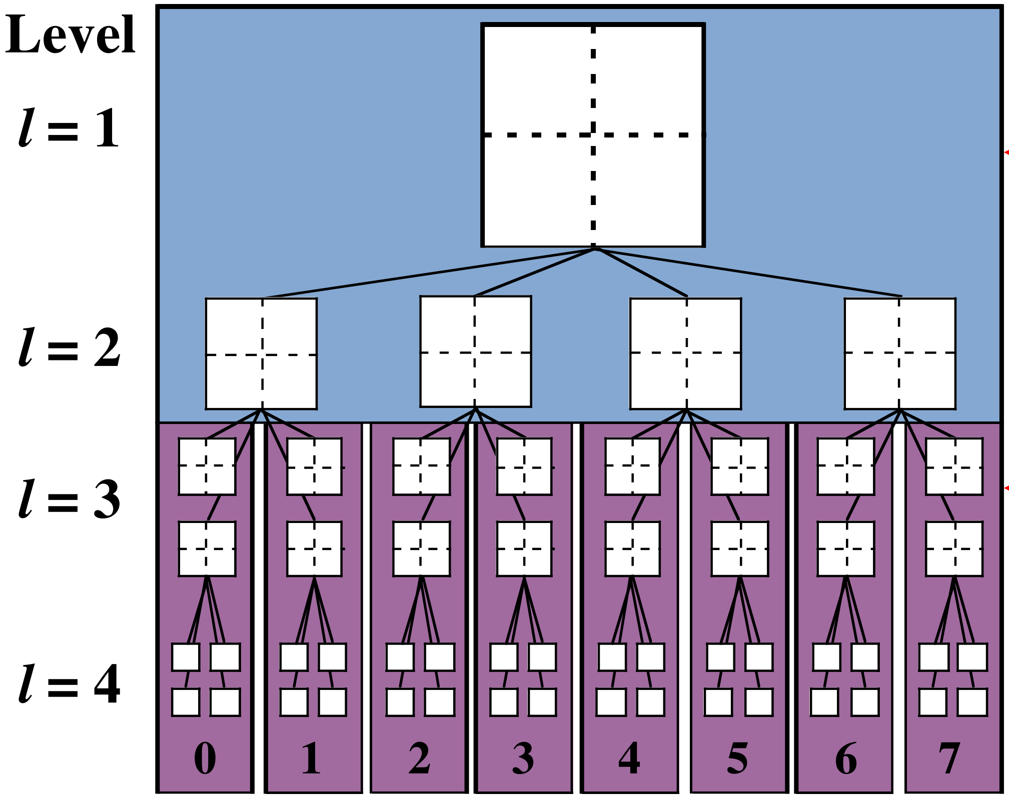
O(*N*) Algorithm



1. **Upward pass computes multipoles for all cells: M-to-M translation**
2. **Downward pass translates multipoles to local terms for all cells**
   - **Constant (189 in 3D) interactive (cousin) cells per destination cell contribute to M-to-L translation** — Within parent's neighbor but not my neighbor
   - **Inheritance from the parent cell: L-to-L translation (& delegation)**
3. **Direct interactions for the nearest-neighbor leaf cells**

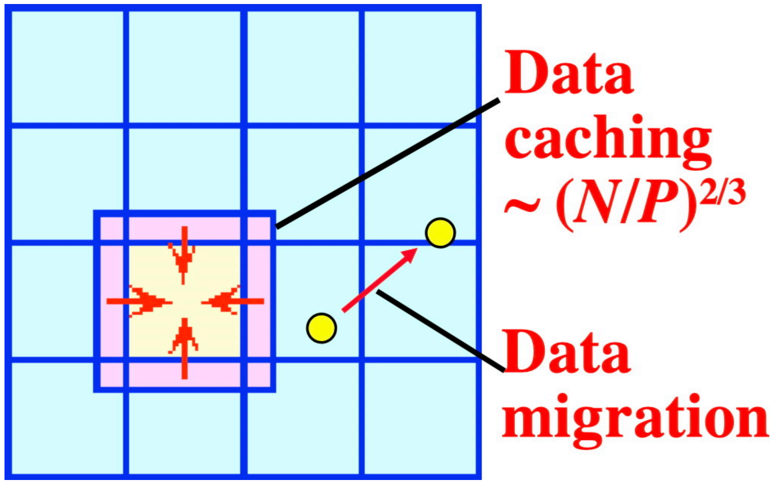
See lecture notes for the MM, ML & LL formula in 2D

Parallel Implementation of FMM



Upper levels:
Global to all processors
Overhead: $O(\log P)$

Lower levels:
Spatial decomposition
Computation: $O(N/P)$

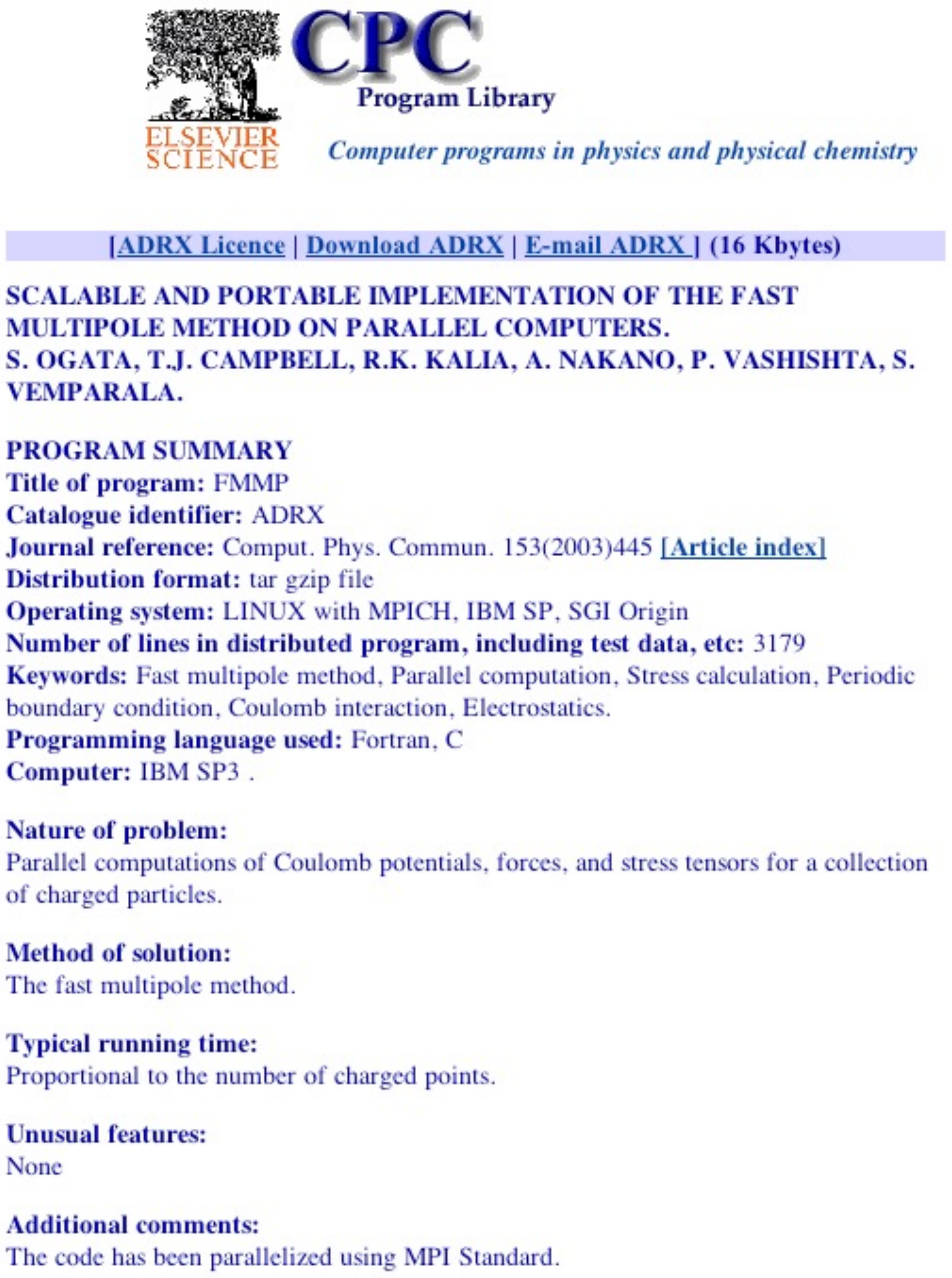


Coarse grain:
$N/P \sim 10^6$; $P \leq 10^3$
$\downarrow$
$N/P \gg \log P, (N/P)^{2/3}$

S. Ogata *et al*., *Comput. Phys. Commun*. **153**, 445 ('03)

https://aiichironakano.github.io/cs653/Ogata-FMM-CPC03.pdf

FMMP Code Dissemination



CPC Program Library

Computer programs in physics and physical chemistry

[ADRX Licence | Download ADRX | E-mail ADRX] (16 Kbytes)

SCALABLE AND PORTABLE IMPLEMENTATION OF THE FAST MULTIPOLE METHOD ON PARALLEL COMPUTERS.
S. OGATA, T.J. CAMPBELL, R.K. KALIA, A. NAKANO, P. VASHISHTA, S. VEMPARALA.

PROGRAM SUMMARY
Title of program: FMMP
Catalogue identifier: ADRX
Journal reference: Comput. Phys. Commun. 153(2003)445 [Article index]
Distribution format: tar gzip file
Operating system: LINUX with MPICH, IBM SP, SGI Origin
Number of lines in distributed program, including test data, etc: 3179
Keywords: Fast multipole method, Parallel computation, Stress calculation, Periodic boundary condition, Coulomb interaction, Electrostatics.
Programming language used: Fortran, C
Computer: IBM SP3 .

Nature of problem:
Parallel computations of Coulomb potentials, forces, and stress tensors for a collection of charged particles.

Method of solution:
The fast multipole method.

Typical running time:
Proportional to the number of charged points.

Unusual features:
None

Additional comments:
The code has been parallelized using MPI Standard.

O(*N*) Psuedo-charge Method for Calculating Stresses in Coulombic Systems

Microscopic stress tensor: $\overleftrightarrow{\pi}_i = \sum_{j \neq i} \vec{r}_{ij} \vec{f}_{ij}^{T} = q_i \sum_{j \neq i} q_j \frac{\vec{r}_{ij} \vec{r}_{ij}^{T}}{{r_{ij}}^3}$

Attach information of particle position to the particle charge:

$$\vec{C}(\vec{r}_i, \vec{k}) = -\sum_{j \neq i} \frac{\vec{r}_{ij}}{{r_{ij}}^3} q_j \exp(i\vec{k} \bullet \vec{r}_j)$$

Stress tensor is obtained by numerical differentiation:

$$\frac{\partial}{\partial k_\alpha} \{C_\beta(\vec{r}_i, \vec{k}) \exp(-i\vec{k} \bullet \vec{r}_i)\} \xrightarrow[k_\alpha \to 0]{} = i \sum_{j \neq i} q_j \frac{r_{ij}^{\alpha} r_{ij}^{\beta}}{{r_{ij}}^3}$$

No need for multipole translation operators for the stress tensor, which are unknown

S. Ogata *et al.*, *Comput. Phys. Commun*. **153**, 445 ('03)

https://aiichironakano.github.io/cs653/Ogata-FMM-CPC03.pdf

Example Run

Random positions in $[0,1]^2$; random charges in [0,1]

```
Npar = 1000: Number of particles
L = 4:       Quadtree height
P = 6:       Number of multipole terms
```

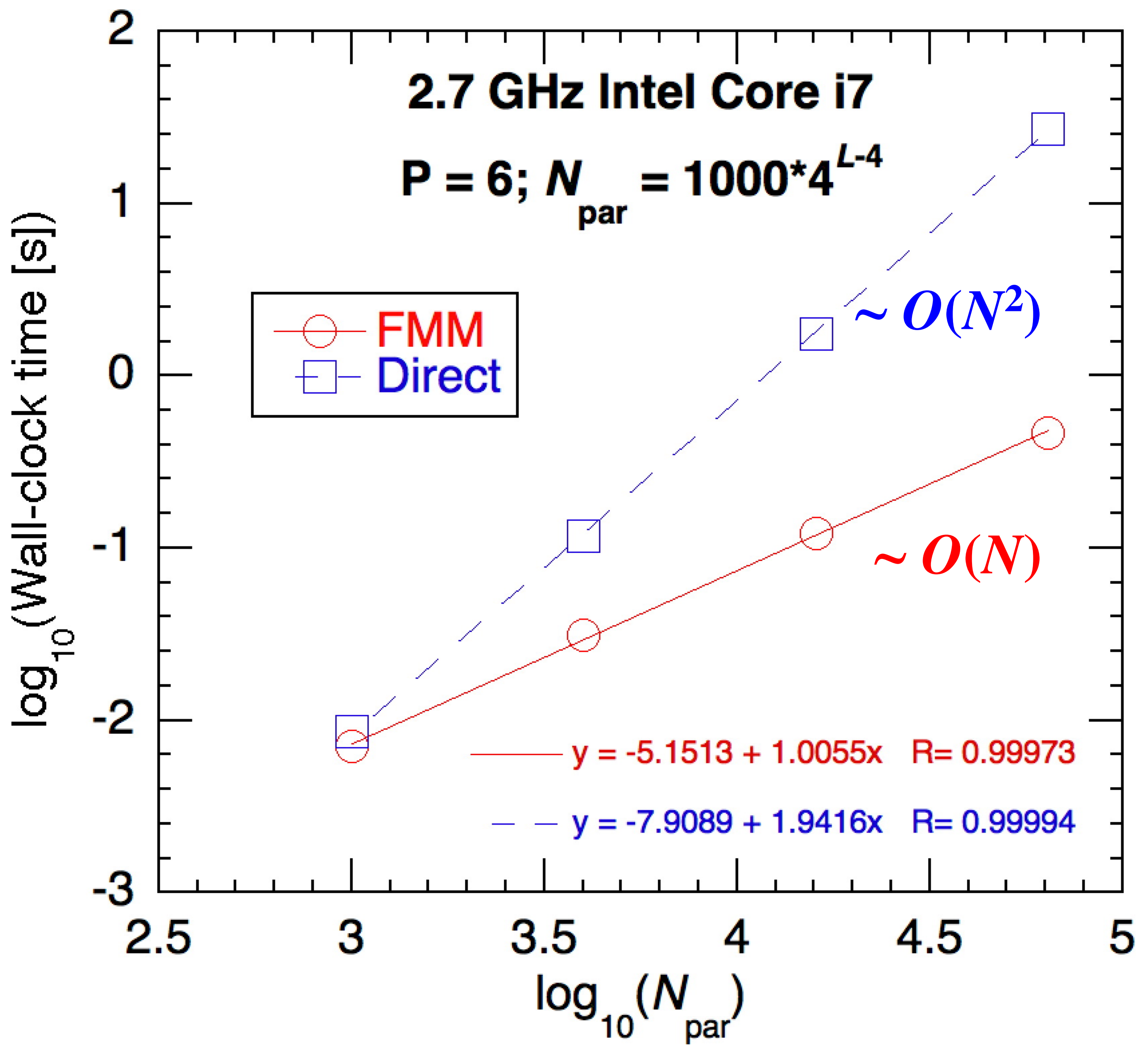
```
$ cc -o fmm2d fmm2d.c -lm
$ ./fmm2d
===== Max potential difference = 5.363006e-06 =====
===== Total FMM vs. direct energies & error =
 -1.008232e+05 -1.008232e+05 -7.271538e-08 =====
===== FMM & direct CPU times = 7.047000e-03 8.541000e-03 =====
```

Try

```
Npar = 4000/16000/64000
L = 5/6/7
P = 6
```

https://aiichironakano.github.io/cs653/src/FMM/

Time-Complexity Measurement



Fast Multipole Method: Bottom Line

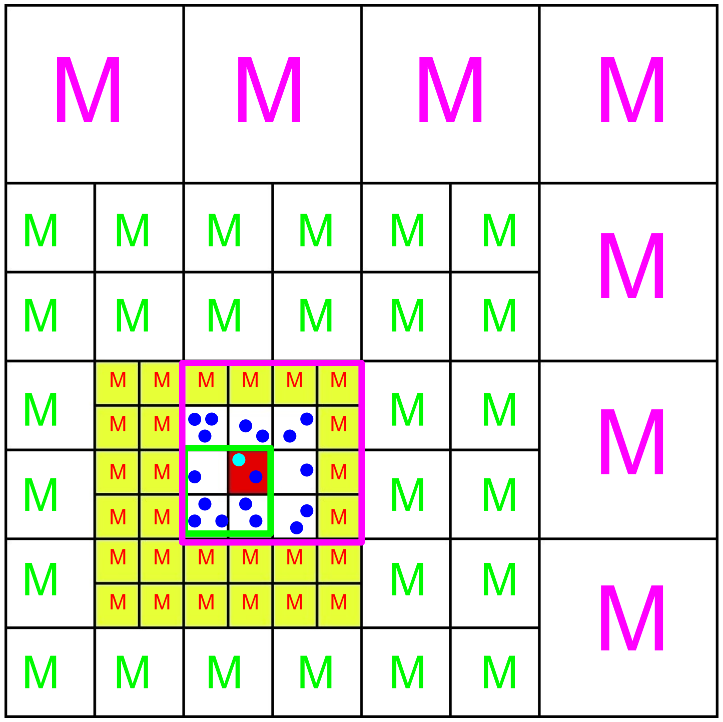
- **FMM algorithm evaluates electrostatic potential $\phi(z_j)$ at all particle positions z_j ($j = 0, \ldots, N_{\text{par}} - 1$).**
- **The whole purpose of the algorithm is to compute the local expansion coefficients, $\Psi_c^{(l)}$, of the electrostatic potential for all destination cells c at the leaf level, $l = L$, which arise from particles in non-nearest-neighbor (n.n.n.) leaf cells; they are computed *via* multipoles $\Phi_c^{(l)}$ of source cells (larger cells are used at upper tree levels for longer distances).**
- **The n.n.n contribution to $\phi(z_j)$ is computed using local expansion, whereas that from the nearest-neighbor (n.n.) leaf cells is computed by direct particle sum.**

n.n.n. contribution *via* multipole-derived local expansion

$$\phi(z_j) = \sum_{\alpha=0}^{P} \Psi_{c(j)}^{(L)}(\alpha)(z_j - z_{c(j)})^{\alpha} + \sum_{\substack{k \in n.n.(c(j)) \\ k \neq j}} q_k \log(z_j - z_k)$$

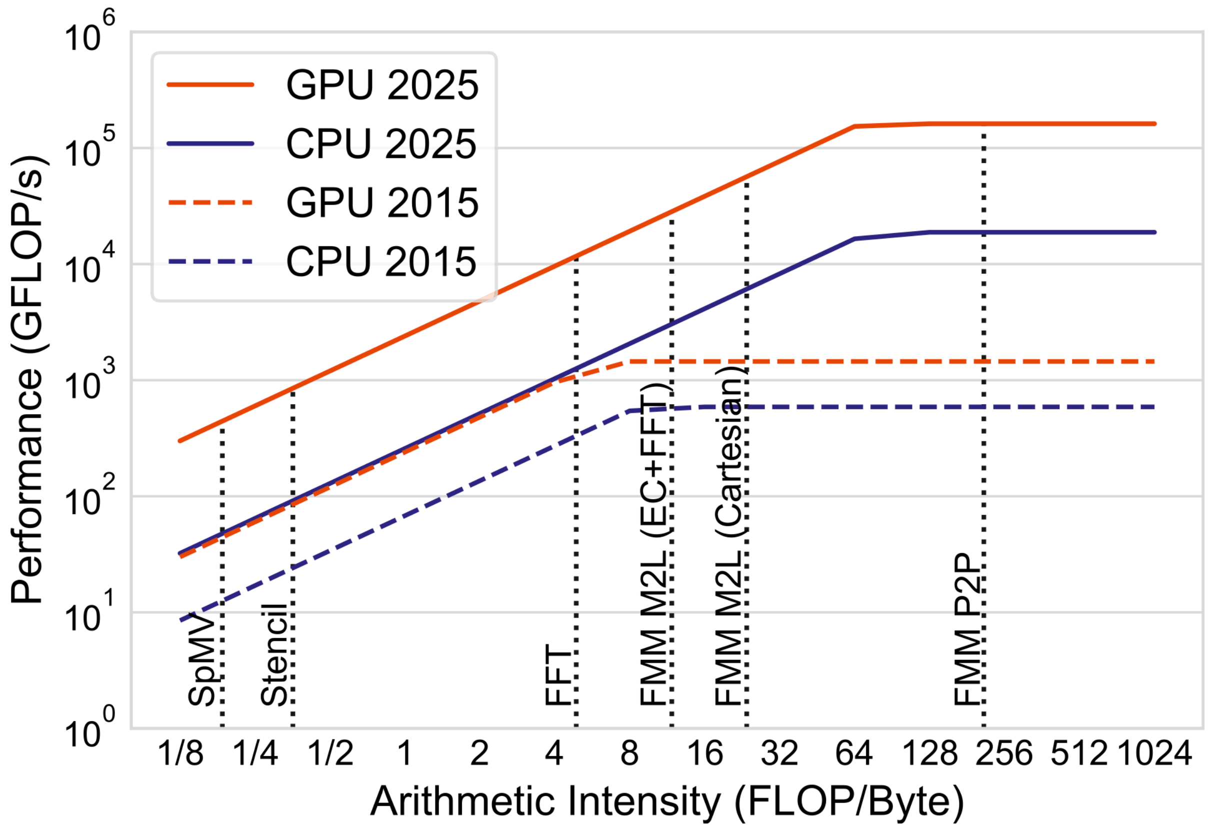
n.n. contribution *via* direct particle sum

P: Truncation order of local expansion
$\Psi_c^{(l)}(\alpha)$: α-th order local expansion coefficient for cell c at level l
$c(j)$: The leaf cell that j-th particle belongs to
$z_{c(j)}$: The center position of cell $c(j)$



Smart *vs*. Brute-force Computations

- **While the tree algorithm drastically reduces the computational complexity, it does not fully utilize computational resources due to its low computation/communication intensity.**
- **Time to re-think computational architectures to embrace smart (sparse) algorithms?**



H. Ibeid *et al.*, *J. Par. Distrib. Comput*. **136**, 63 ('20)