GPU Computing with CUDA
Lecture 10 - Applications - N body problem

Christopher Cooper
Boston University

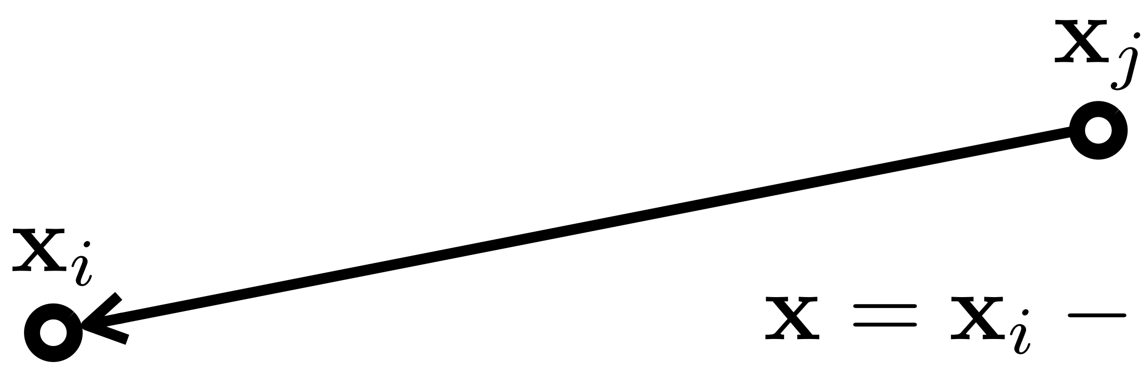
August, 2011
UTFSM, Valparaíso, Chile

Outline of lecture

- What is an N body problem?
- What kind of problems can be solved with N body approach?
- Fast calculation algorithms
- N-body problem on GPU

Introduction

- N body problem calculates the interaction between N bodies



$$\mathbf{x} = \mathbf{x}_i - \mathbf{x}_j$$

$$|\mathbf{x}|_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

- Classic example: Gravitational Potential

$$\phi_i = \sum_{j=0}^{N} \frac{m_j}{|\mathbf{x}|_{ij}} \qquad \nabla\phi = g$$

Introduction

- Useful for anything with Green's function!

- Poisson $\nabla^2\phi = f$
  - Astrophysics
  - Fluid mechanics
  - Electrostatics

- Helmholtz $\nabla^2\phi + k^2\phi = f$
  - Acoustics
  - Electromagnetics

- Poisson- Boltzmann $\nabla(\epsilon\nabla\phi) + k^2\phi = f$
  - Geophysics
  - Biophysics

$$\int_\Omega G\nabla^2\phi d\Omega = \int_\Omega G f d\Omega$$

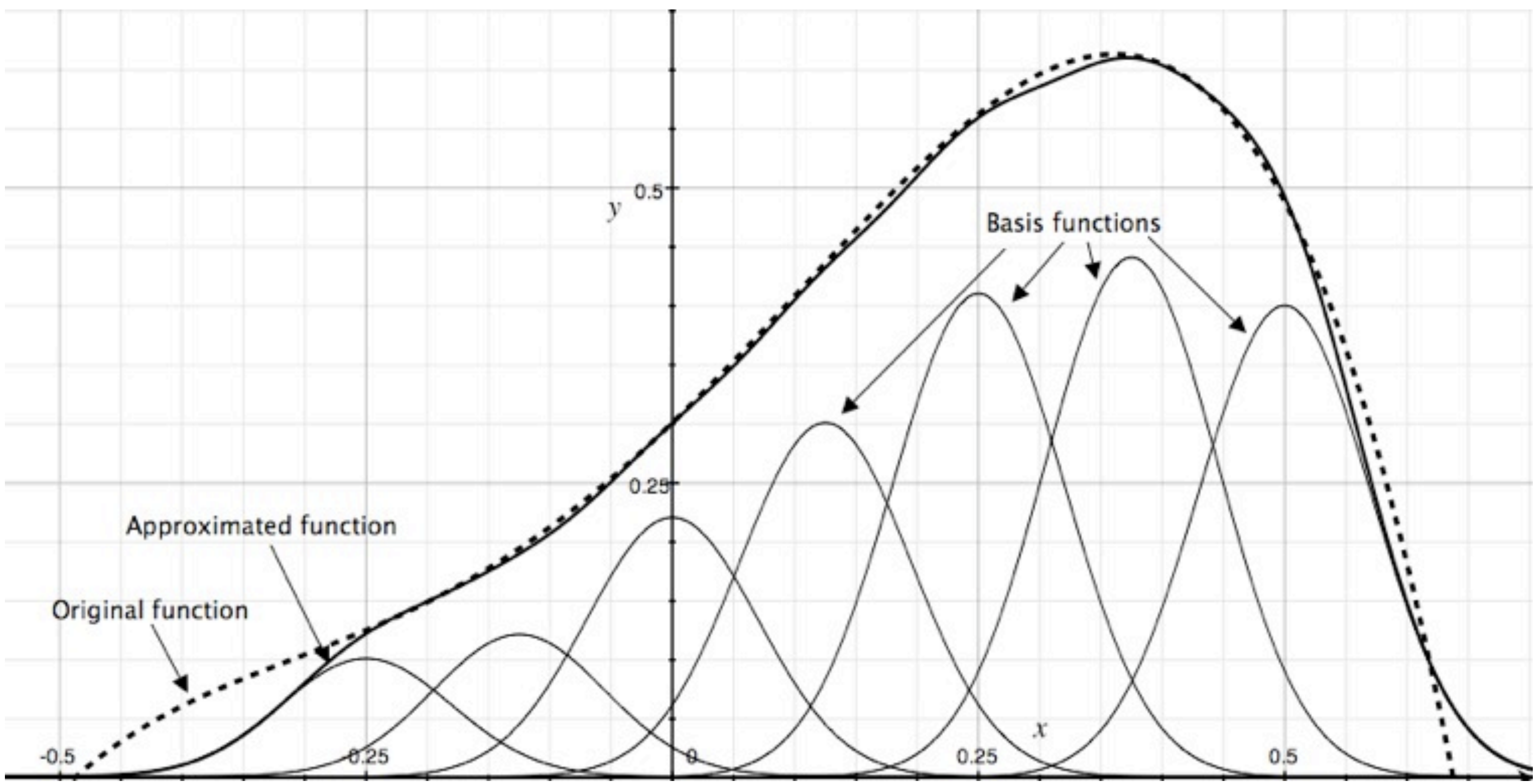
$$\int_\Omega \phi\nabla^2 G d\Omega + \int_\Gamma \mathbf{n}\cdot[G\nabla\phi - (\nabla G)\phi]d\Gamma = \int_\Omega G f d\Omega$$

$$\phi = \int_\Omega G f d\Omega$$

Introduction

- Radial Basis Function interpolation

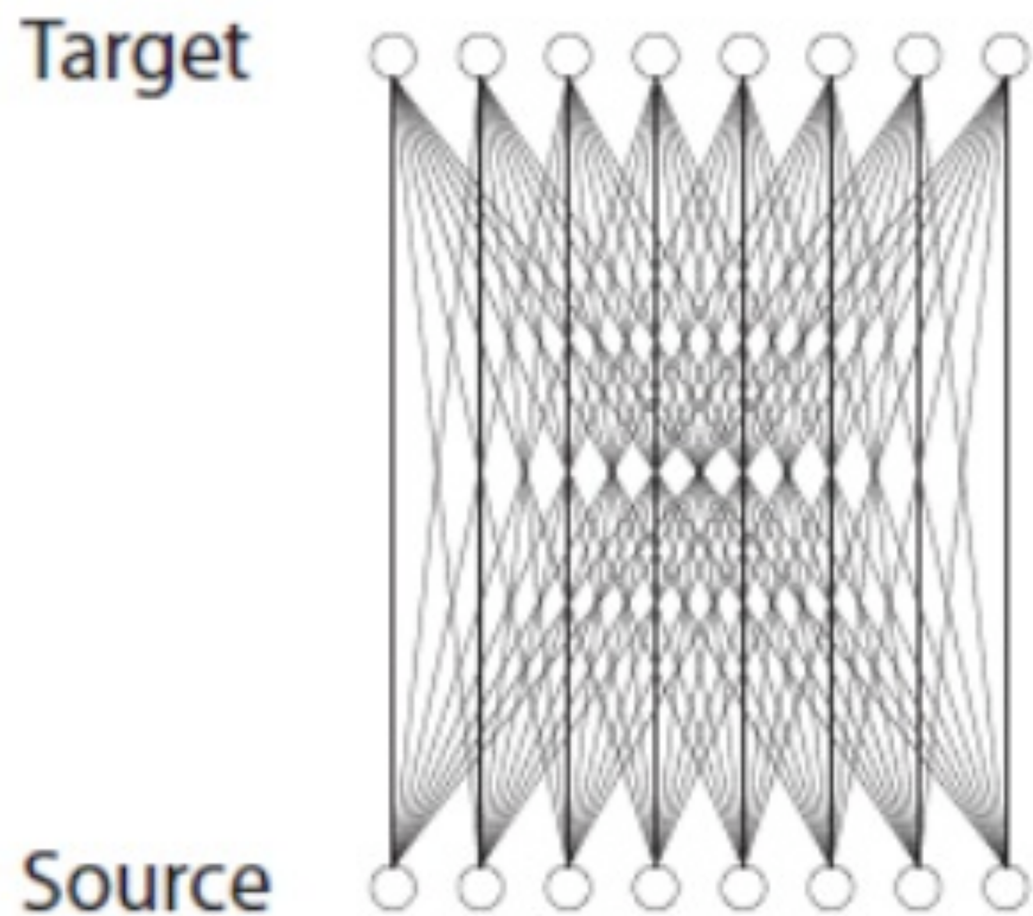
$$u(\mathbf{x}) = \sum_{i=0}^{N} \alpha_i \phi(|\mathbf{x} - \mathbf{x}_i|)$$



Order of N body problems

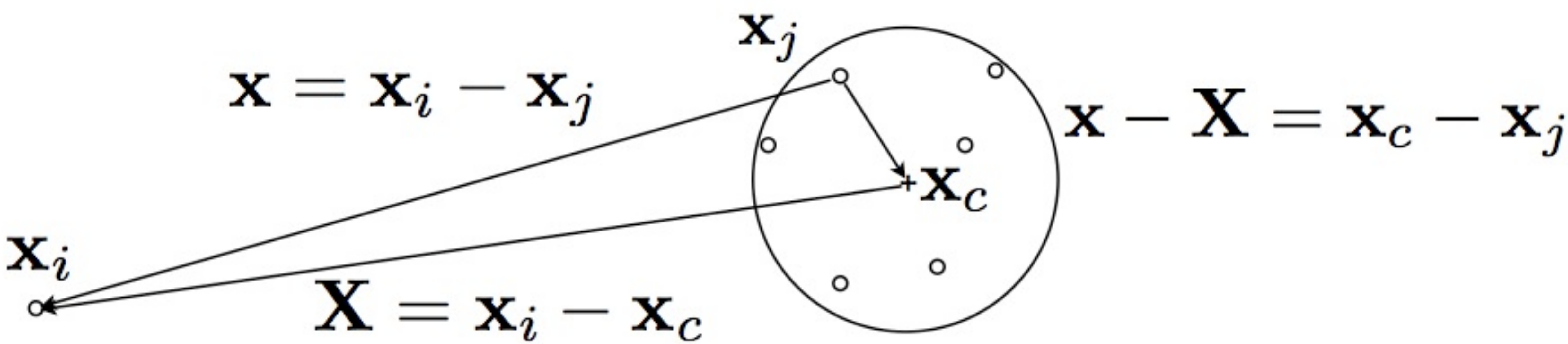
- Direct calculation is O(N²)

$$\Phi_i = \sum_{j=0}^{N} \frac{m_j}{|\mathbf{x}|_{ij}}$$



Fast algorithms

- Multipole expansions

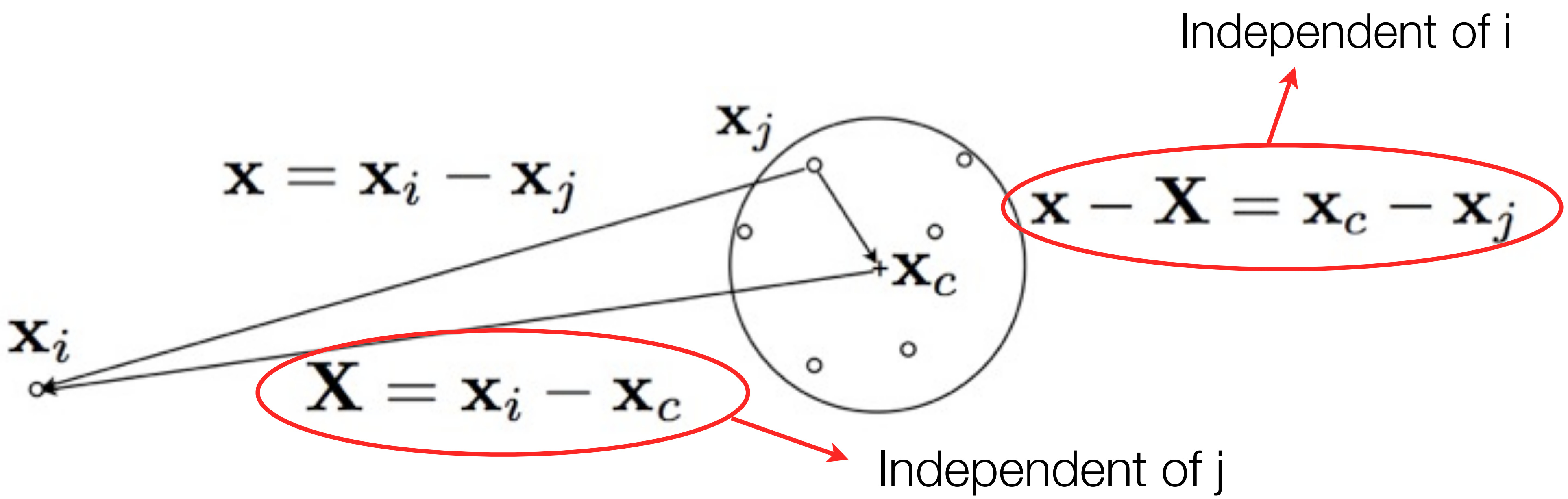


$$f(\mathbf{x}) = f(\mathbf{X}) + (\mathbf{x} - \mathbf{X})\frac{f'(\mathbf{X})}{1!} + (\mathbf{x} - \mathbf{X})^2\frac{f''(\mathbf{X})}{2!} + \ldots$$

$$\frac{1}{r} = \sum_{n=0}^{p} \frac{1}{n!}(\mathbf{x} - \mathbf{X})^n \frac{\partial^{(n)}}{\partial \mathbf{X}} \frac{1}{R}$$

$$R = \sqrt{X^2 + Y^2 + Z^2}$$

Fast algorithms

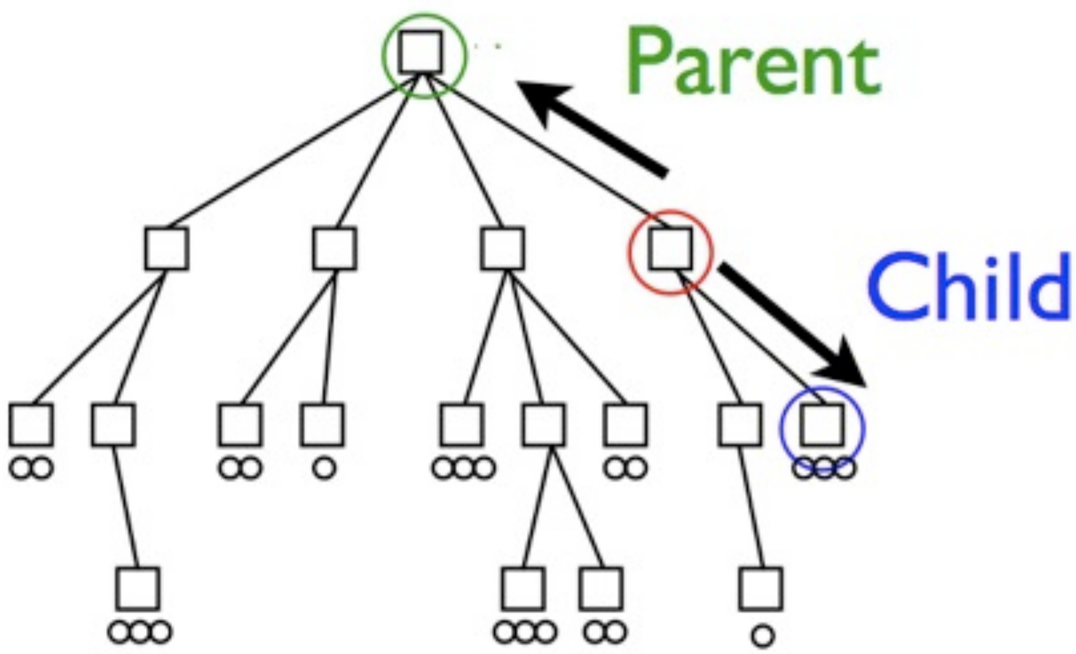


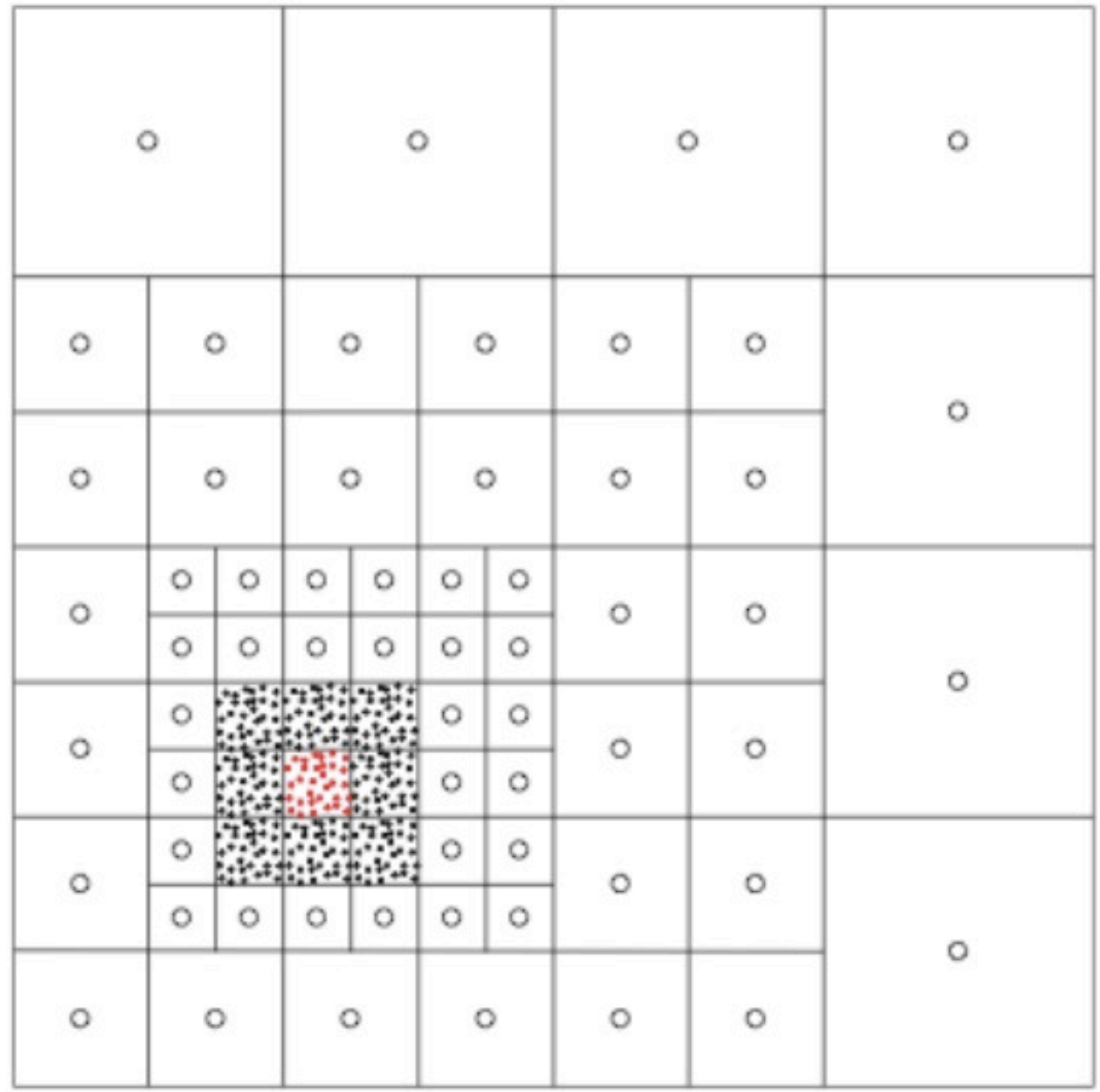
$$\frac{1}{r} = \sum_{n=0}^{p} \frac{1}{n!} (\mathbf{x} - \mathbf{X})^n \frac{\partial^{(n)}}{\partial \mathbf{X}} \frac{1}{R} \qquad R = \sqrt{X^2 + Y^2 + Z^2}$$

$$\Phi_i = \sum_{j=0}^{N} \frac{m_j}{r} = \sum_{j=0}^{N} m_j \sum_{n=0}^{p} \frac{1}{n!} (\mathbf{x} - \mathbf{X})^n \frac{\partial^{(n)}}{\partial \mathbf{X}} \frac{1}{R}$$

$$= \sum_{n=0}^{p} \frac{\partial^{(n)}}{\partial \mathbf{X}} \frac{1}{R} \underbrace{\sum_{j=0}^{N} \frac{1}{n!} m_j (\mathbf{x} - \mathbf{X})^n}_{Multipole}$$

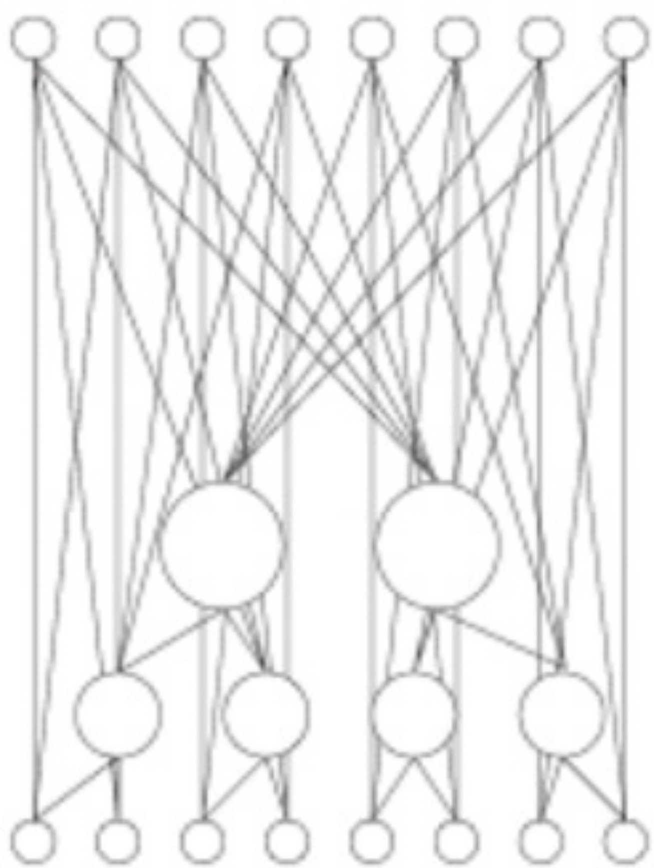
Fast algorithms



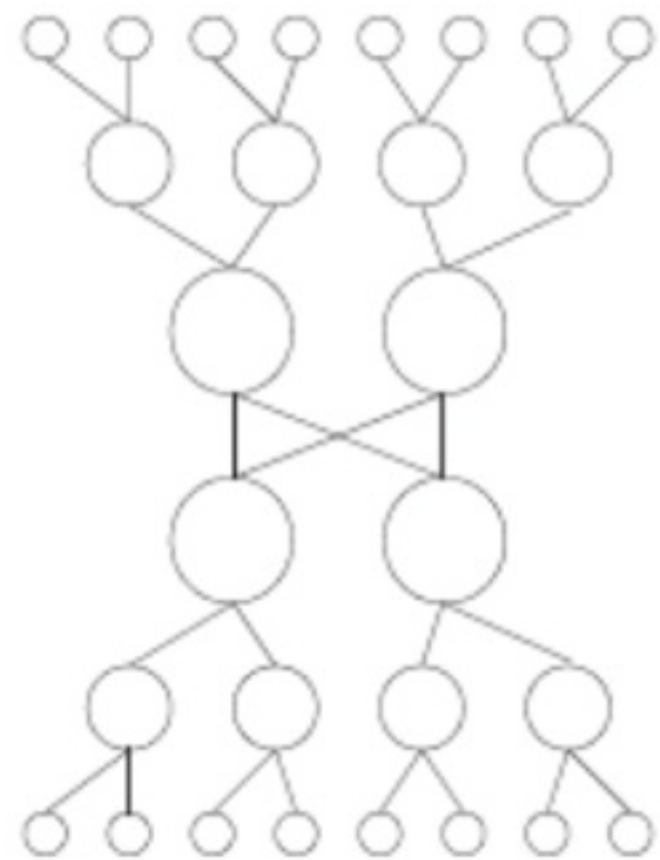


Fast algorithms

- Tree Code - Barnes and Hut
- Fast Multipole Methods (FMM) - Greengard and Rokhlin



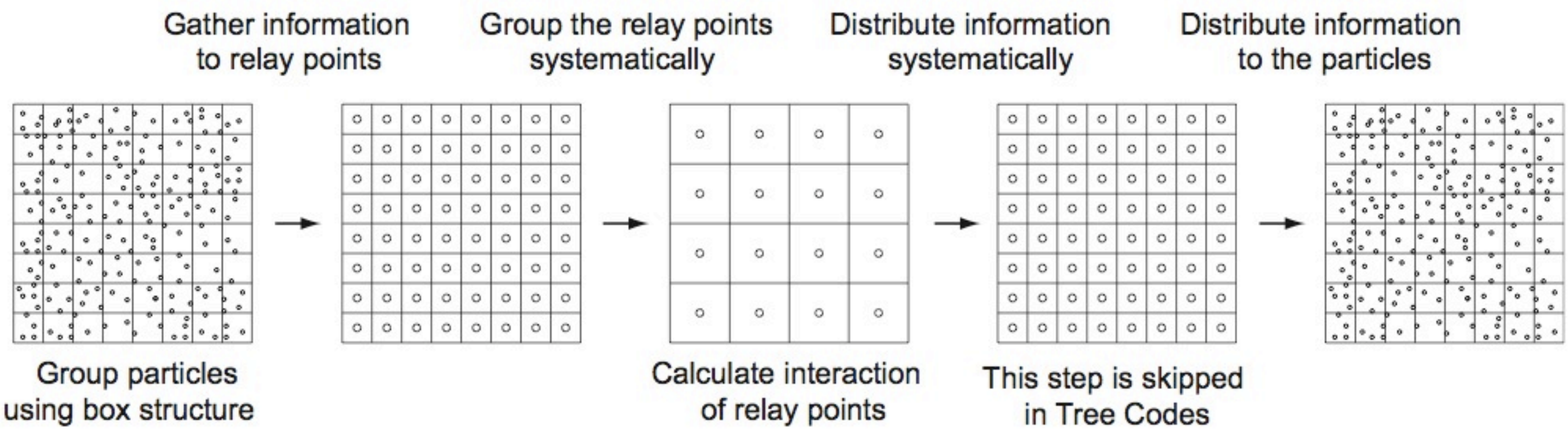
Tree code



FMM

Fast algorithms

- Flow of calculation



N body on GPU

- Tomorrow's lab: implement $O(N^2)$ N body calculation on GPU
- Nyland L., Harris M., Prins J. "Fast N-Body Simulation with CUDA". GPU Gems 3, Chapter 31.
- Lots of operations per load
- Much faster, but still $O(N^2)$!

N body on GPU

- Look at the problem as a matrix vector product

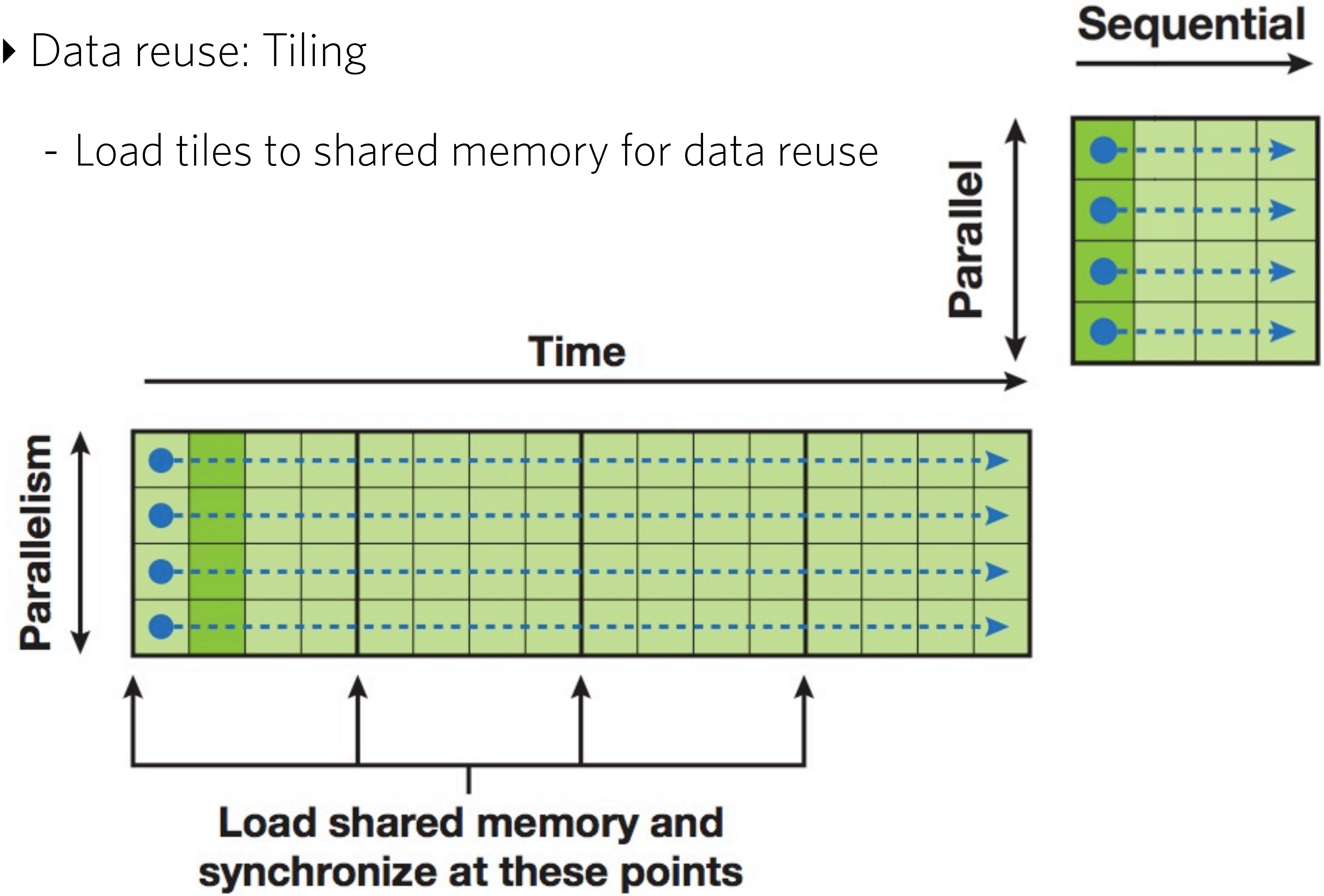
$$\Phi_i = \sum_{j=0}^{N} \frac{m_j}{|\mathbf{x}|_{ij}}$$

$$\Phi_i = \sum_{j=0}^{N} \frac{m_j}{(|\mathbf{x}|_{ij} + \epsilon^2)}$$

- Each thread will compute one row (not one element)

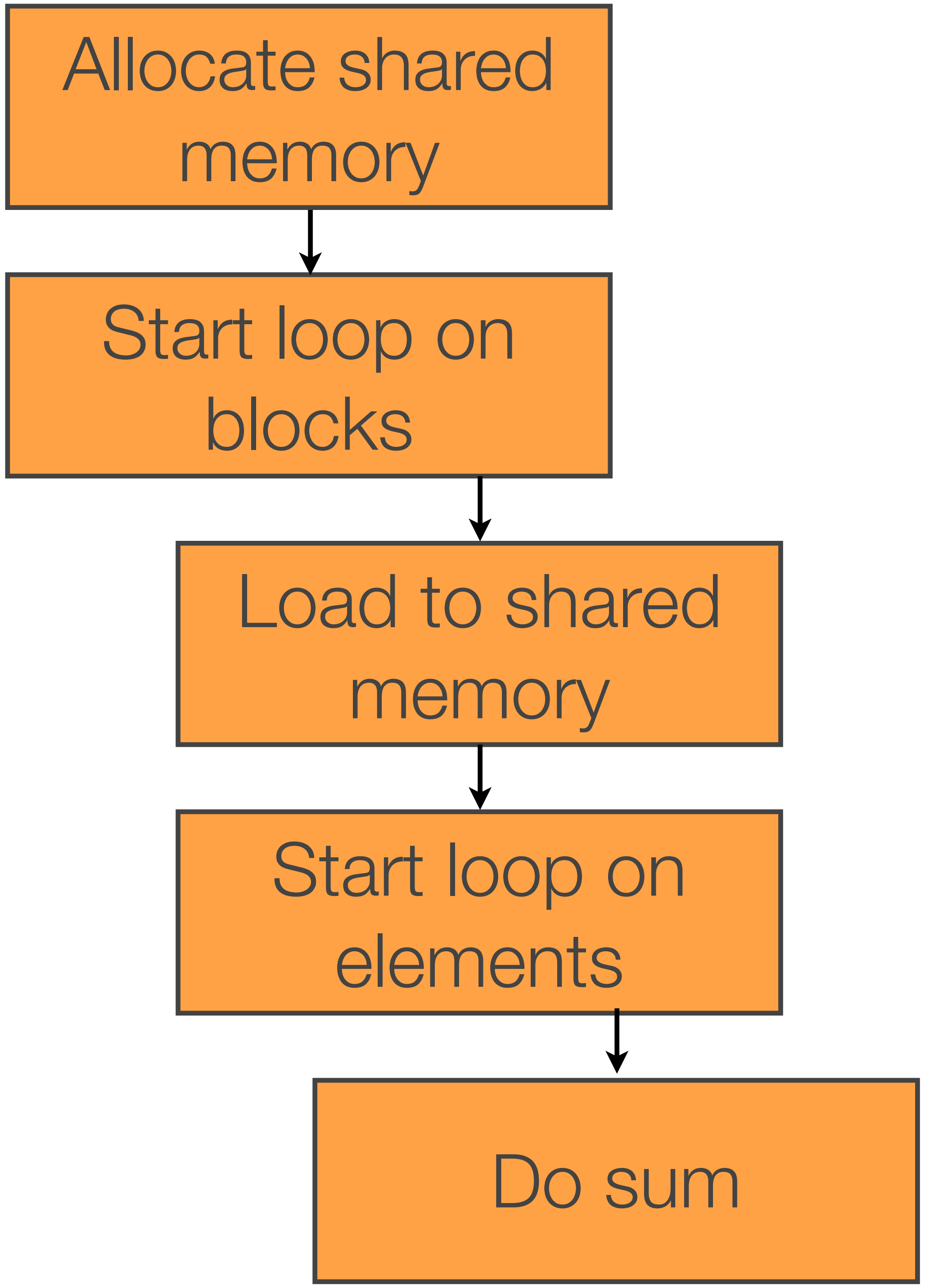
N body on GPU

- Data reuse: Tiling
  - Load tiles to shared memory for data reuse



N body on GPU

- Kernel



N body on GPU - Optimization

- Loop unrolling
  - Avoid unnecessary operation
  - `#pragma unroll 32`
- Separate last loop
  - The number of elements might not be multiple of the block size
  - Separate last loop to avoid unnecessary warps performing a calculation
- Vary block size

N body on GPU - Performance metric

- Compute bounded problem
  - Performance in FLOPS/s
  - Count number of floating point operations and divide by kernel execution time