N-body simulation

Part 3: parallelisation

epcc

THE UNIVERSITY OF EDINBURGH

Overview

- Parallelisation challenge for N-body simulation
- Possible parallelisation strategies
  - Task farm
  - Replicated data
  - Particle-based decomposition
  - Force-based decomposition
  - Domain decomposition
- Parallelising long-range forces
  - Particle-Mesh Ewald (PME)
  - Barnes-Hut Tree
- Common N-body simulation codes

N-body algorithms – where is the work?

- Typically not memory intensive
  - Often mainly vectors of particle information stored
  - Memory can become a bottleneck for cosmological simulations (many billions of particles)
- N-body simulations ‘large’ in two ways:
  - Number of particles
  - Number of timesteps
- Most parallelisation effort focused on force calculations
  - Treat short-range & long-range differently
  - Approach depends on long-range force calculation scheme
  - Impacts all aspects of N-body algorithm
- Parallel I/O also implemented in some packages

Parallelisation challenges

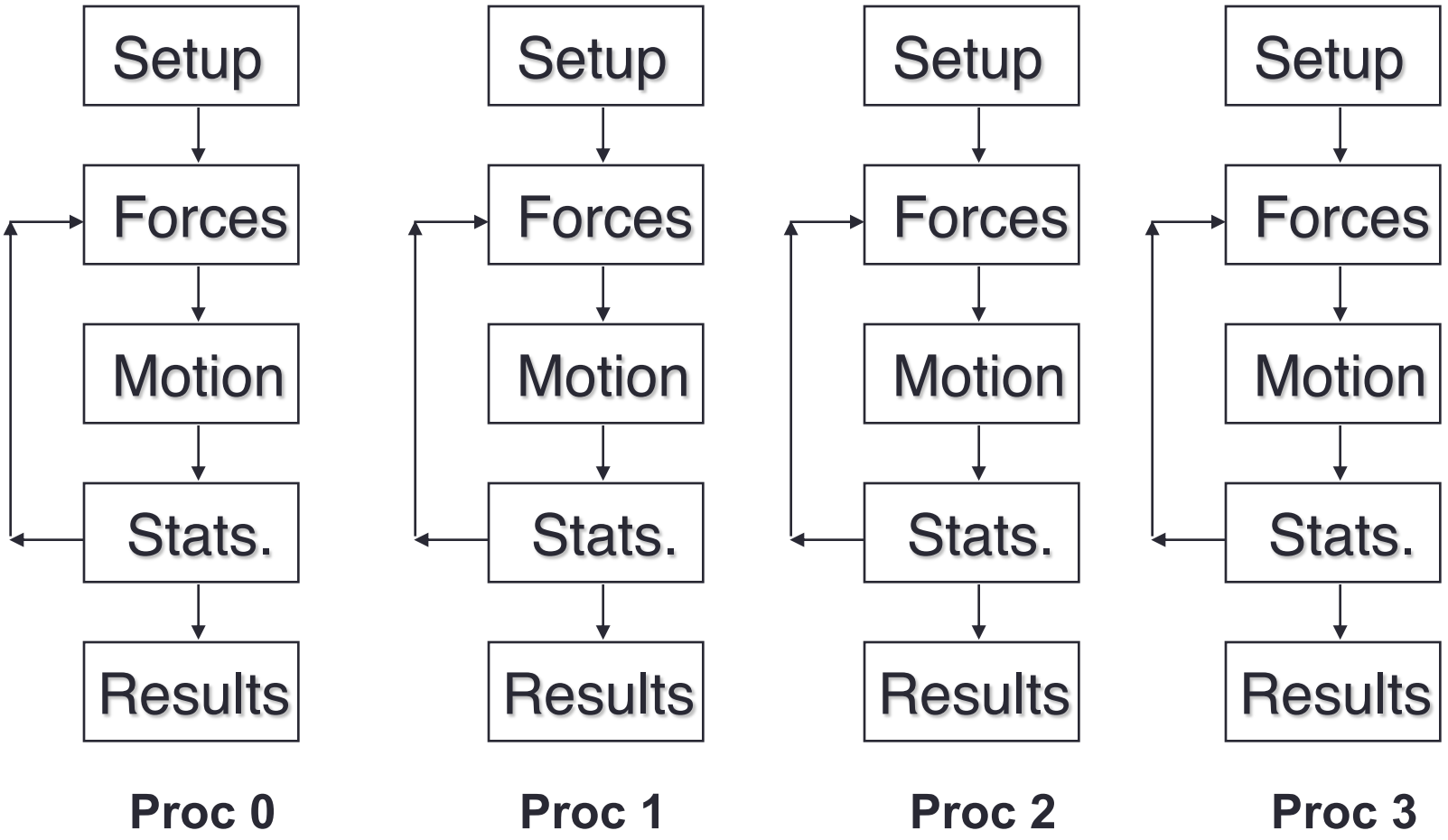
- Load Balancing:
  - Share work equally between processes
  - Share memory requirements equally
  - Maximum concurrent use of each core
- Communication vs computation:
  - Aim to overlap communication with computation
  - Prefer asynchronous communication
    - But most time integration schemes force synchronisation after each step
  - Prefer local communications / direct memory access to global comms.
- Make use of vectorisation

Possible N-body parallelisation strategies

What parallelisation approaches are available to us?

1. Task farm
2. Replicated data
3. Particle-based decomposition
4. Force-based decomposition
5. Spatial domain-based decomposition

Simple task farm



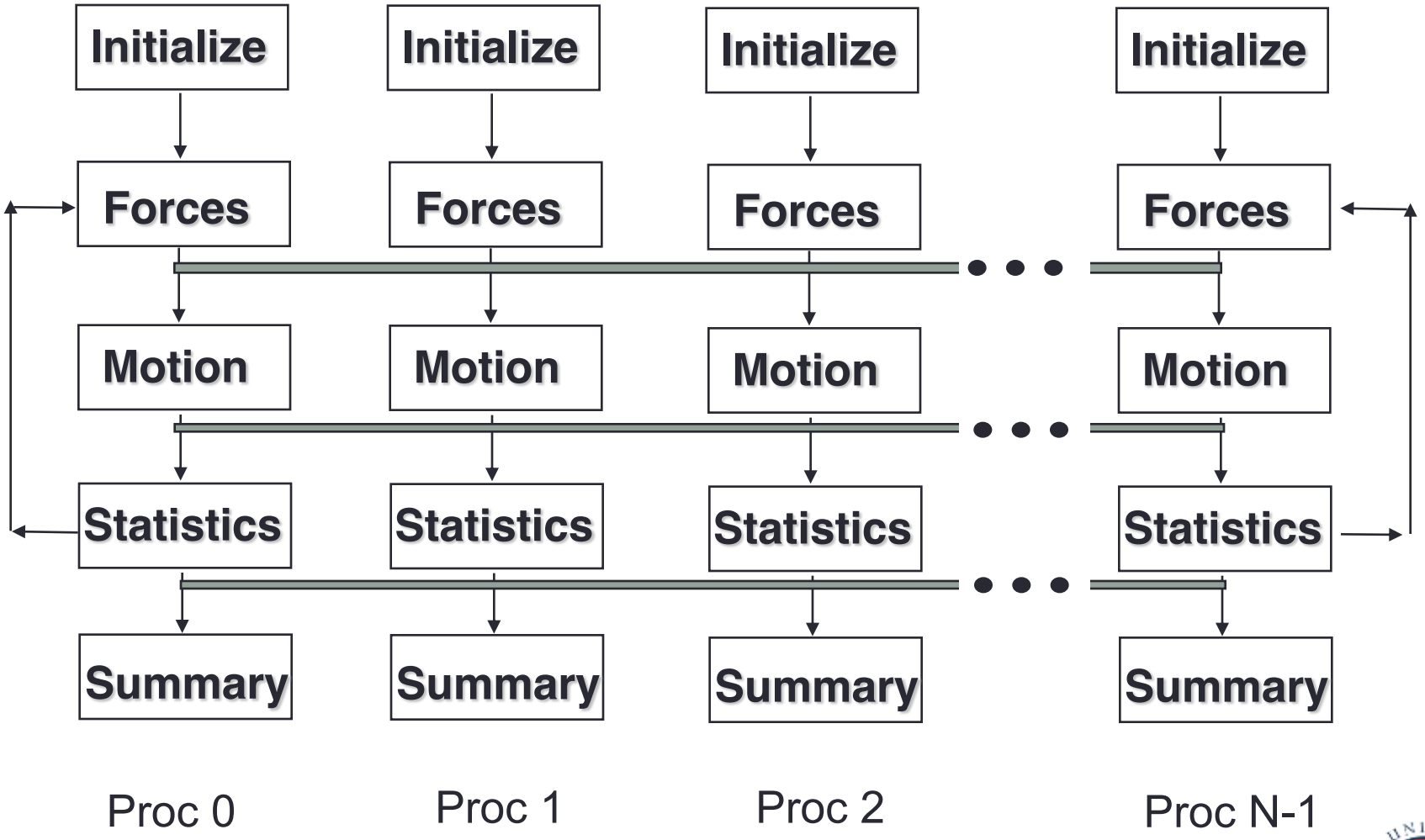
epcc

THE UNIVERSITY OF EDINBURGH

Simple task farm

- Advantages:
  - Simple to implement - no communications
  - Maximum parallel efficiency – excellent throughput
  - Perfect load balancing (assuming very similar tasks)
  - Good scaling behaviour (as long as you have enough tasks to get done)
  - Suitable method for many independent simulations
    - e.g. ensemble of similar systems, for statistical sampling
- Disadvantages:
  - Complexity of each individual simulation task severely limited:
    - very short timescale dynamics
    - very small problem size (N)
    - very simple force calculations

Replicated data



epcc

THE UNIVERSITY OF EDINBURGH

Replicated data

- Advantages:
  - Simple to implement (every process has a view of all particles)
  - Good (dynamic) load balancing possible
  - Can handle complex force fields
    - use more cores → fewer particles per process
  - Good scaling with N
    - but only until (per-process) memory limit is reached
- Disadvantages:
  - Very high communication overhead
    - Every process must integrate updates from all other processes after each step
  - Limited scaling with process/core count (due to communications)
  - Very large memory requirement
  - Unsuitable for massive parallelism

Particle-based decomposition

- Each process assigned a **subset of particles**
  - Computes interactions between that subset and all other particles
- Particles assigned to a process might not be spatially close
- P processes → each assigned N/P particles
- Consider matrix of pairwise forces F(i,j)
  - each process assigned N/P **rows** of the force matrix to compute
- Each process updates positions and velocities of "its" particles, no matter where they move

Particle-based decomposition

- For process P_z with elements F_z, many more particle positions than those stored by process P_z needed for force calculation
- **Each timestep: each process must receive updated particle positions from all other processes (all-to-all communication)**
- Best case algorithms:
  - $\log_2(P)$ sends and receives
  - exchanges of O(N) data values

Force-based decomposition

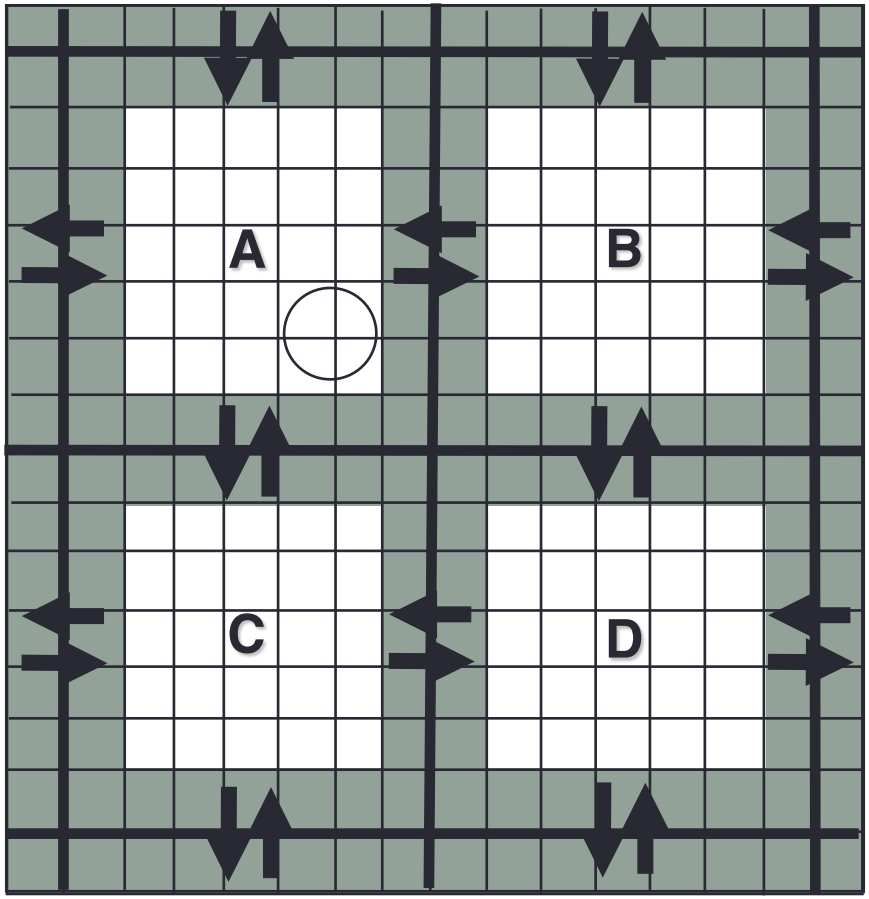
- Each process assigned a **subset of interactions** (**forces)**
  - Computes **all** interactions between particles in a particular subset
- Good load balance (decomposing based on work)
- Block-per-process decomposition of force matrix, not row-per-process decomposition as for particle-based
  - Each process assigned a **block** of N^2/P forces to compute
- Less communication than particle-based
  - Still need to accumulate total force on each particle across blocks
  - Each process needs to compute or obtain updated particles
- Can be hard to implement – require prior knowledge of interactions, need to update scheme if these change

Domain decomposition

- Each process assigned a **spatial region** (**domain**)
  - Computes total forces on all N/P particles in its domain for each timestep, updates their positions & velocities
- Can compute *short-range forces* on particles in its domain with minimal communication (direct neighbour domains only)
  - need to know about particles within short range of domain boundaries
- Can accommodate different schemes for computation of *long-range forces*
- Many variants, most flexible and best scaling, but can be complex to implement!

Domain decomposition

2D Example



- Decompose particles into domains
- Map domains onto processes
- Use *link/halo/ghost cells* in each domain
- Communicate halo cells between adjacent processes
- Compute forces
  - Cut off (ignore) short-range interactions beyond a short-range cutoff $r_{cut} << L_{cell}$
- Solve equations of motion (integrate), updating positions, velocities
- Re-allocate any particles leaving domains

epcc

THE UNIVERSITY OF EDINBURGH

Domain decomposition

- Advantages:
  - Predominantly local communications for *short-range forces*
    - Can accommodate different *long-range* computation schemes
  - Inherently good load balancing if system is isotropic (homogeneous)!
    - Dynamic load balancing possible (adaptive domain size)
  - Good scaling with large N
  - Maps well onto hybrid distributed + shared memory architectures
    - e.g. MPI for periodic communication between domains, OpenMP threads for short-range interactions within domain
- Disadvantages
  - Sub-optimal scaling with processes for small N
  - Requires short-range potential cut off $\leq$ subdomain size
  - Optimal communication may be complex to design & implement

Parallelising long-range forces

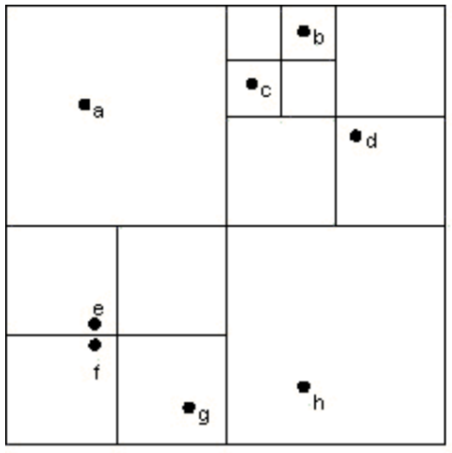
Periodic boundary conditions in parallel: Particle-Mesh Ewald (PME)

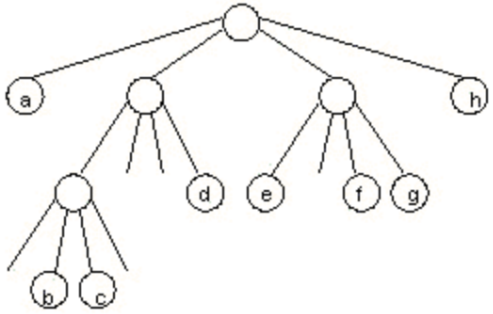
- Serially, compute:
  - Short-range forces by direct sum over all close particle pairs
  - Long-range mesh-based potential $\Phi_{\mathbf{lr}}(\vec{r})$ felt by all particles, using FFTs
- In parallel (domain decomposition):
  - Each process computes short-range forces within its domain
  - Long-range interactions span across domains
    - Each process computes mesh-based $\rho(\vec{r})$ for its domain
    - Perform FFT to compute $\hat{\rho}(\vec{k})$ in Fourier space
      - Gather mesh densities from all domains into one process first
    - Compute $\Phi_{\mathbf{lr}}(\vec{r})$ by taking the inverse FFT
      - Distribute mesh potential result to all domains (just the part relevant to each domain)
- Perform FFTs in parallel using FFT library

epcc

THE UNIVERSITY OF EDINBURGH

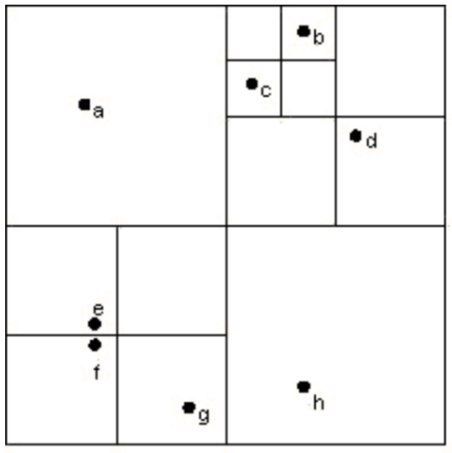
Parallel Barnes-Hut – tree construction

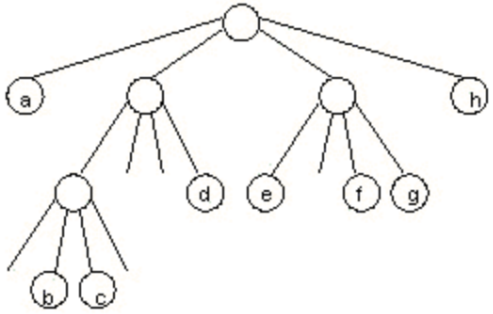




- Each process constructs branch of the Barnes-Hut tree storing cluster representations of particles in its domain
  - Each domain = many nested subdividing boxes (many particles)
- Unless spatial distribution of particles very homogenous, tree not perfectly balanced
  - Create different domain sizes so each process has roughly same number of particles and hence tree nodes

Parallel Barnes-Hut – force computation





- Each process computes forces for particles in its domain:
  - Depth-first traversal, starting with root node (!)
  - Need cluster representations of the other domains
  - Initial all-to-all communication of top-level nodes between processes
- Very rough approximation: no further communication needed
- Higher accuracy: request deeper nodes (higher resolution cluster representations) from specific processes

epcc

THE UNIVERSITY OF EDINBURGH

HPC N-body simulation codes

- Most major N-body codes use forms of **spatial decomposition** using **MPI** to run on distributed-memory HPC machines
- Lot of effort on efficient GPU offloading
- HPC atomistic / Molecular Dynamics simulation codes:
  - AMBER, GROMACS, NAMD, DL_POLY, LAMMPS
- HPC astrophysical / gravitational simulation codes:
  - GADGET-4, Bonsai, GOTHIC, GreeM

Summary

- Parallelisation challenge for N-body simulation
- Possible parallelisation strategies
  - Task farm
  - Replicated data
  - Particle-based decomposition
  - Force-based decomposition
  - Domain decomposition
- Parallelising long-range forces
  - Particle-mesh Ewald (PME)
  - Barnes-Hut Tree
- Common N-body simulation codes