

Solving the N-body problem using GPUs

Scott Wales, Sydney Institute for Astronomy

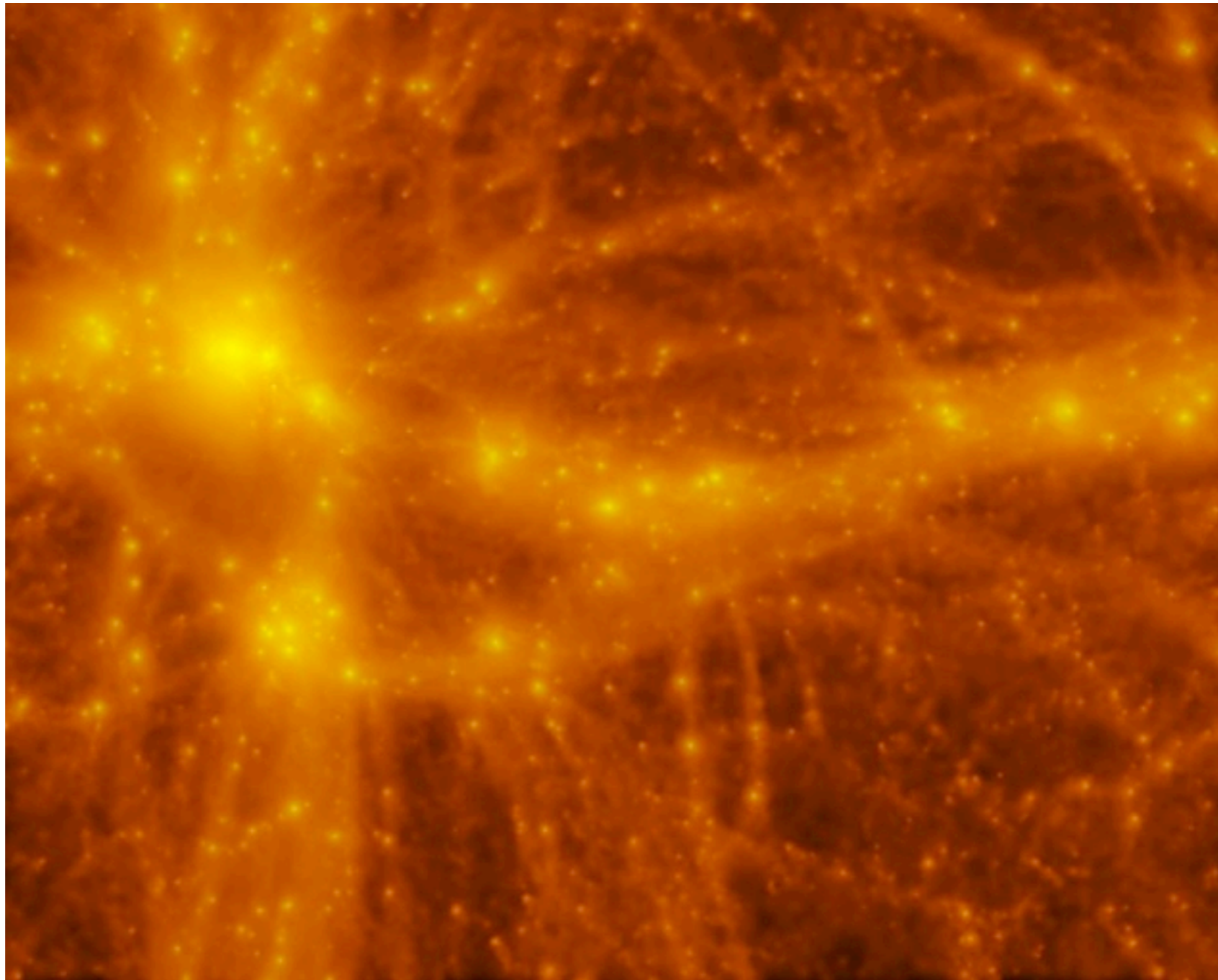
wales@physics.usyd.edu.au



THE UNIVERSITY OF
SYDNEY



The N-Body Problem



Solving The N-body Problem Using GPUs

Scott Wales March 2010

The N-Body Problem

Given a set of N particles with masses $m_1..m_N$, what is the gravitational force felt by each?

A History of N-body Simulations

- Particle-Particle
- Particle-Mesh
- Tree
- Hybrid

Particle-Particle

- For each particle look at every other particle and find the force according to

$$\vec{F}_i = \sum_j \frac{Gm_i m_j \hat{r}_{ij}}{|\vec{r}_{ij}|^2}$$

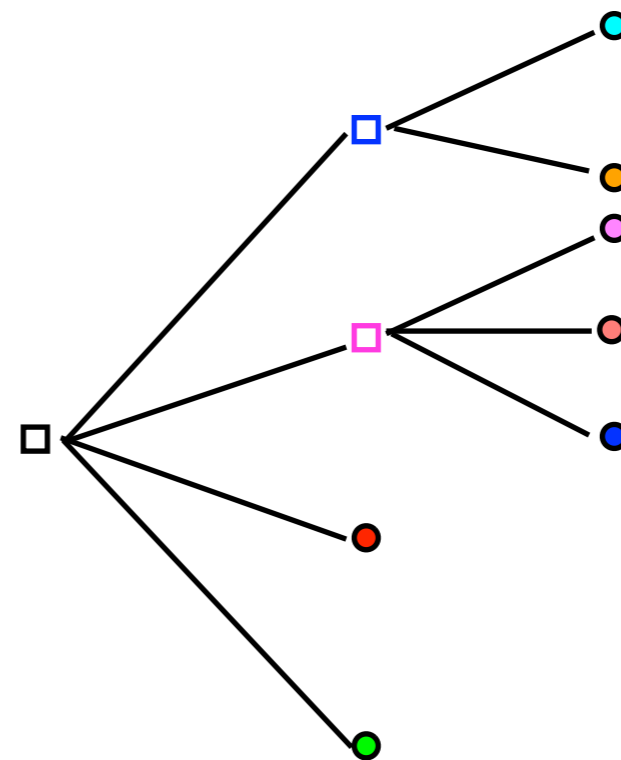
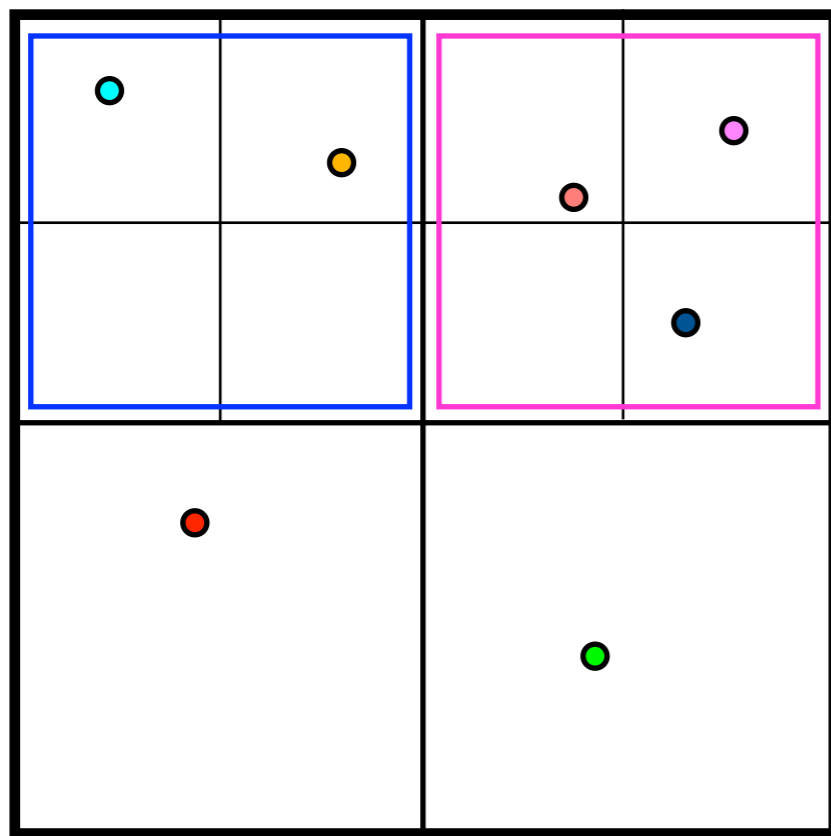
- Scales as $O(n^2)$ - twice as many particles means four times the calculations

Particle-Mesh

- Put the particles on a grid, then solve the Poisson equation in Fourier space and interpolate back from the grid to the particles
- A fast Fourier transform scales as $O(n \log n)$
- Not good for resolutions below the grid scale

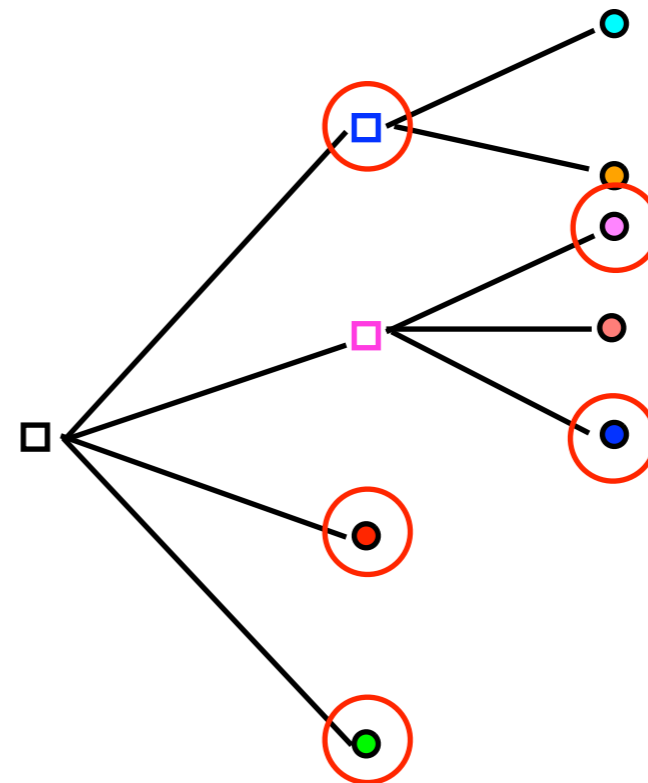
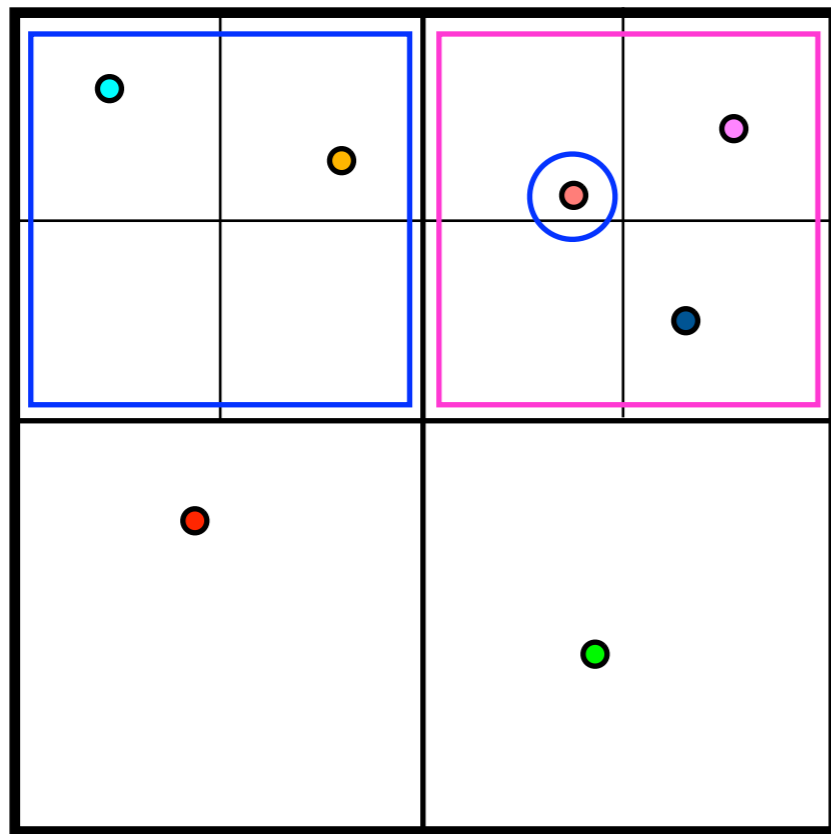
Tree

- Divide the domain into separate regions
- Create a tree from these regions



Tree

- Use moments at long range, particles at short range



Hybrid

- Use an accurate method at short range and a fast method at long range
- Particle-Particle/Particle-Mesh
- Tree/Particle-Mesh

GPUs?

- Want to do the same operation over a lot of data
- Easiest method? Particle-Particle

Is n^2 Good Enough?

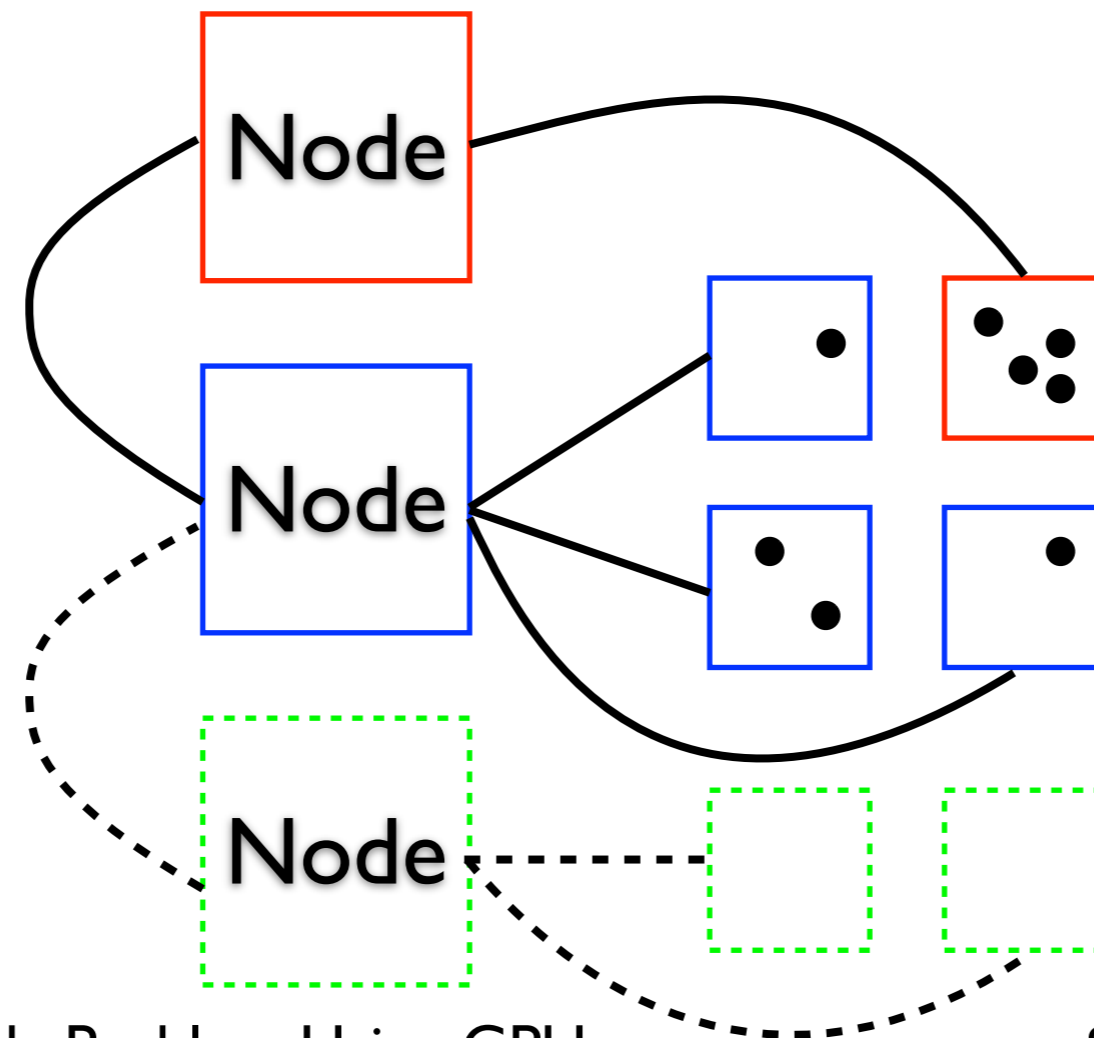
- Care more about simulation size than speed
- Can we make a smarter algorithm?

Hybridise

- Short range forces on the gpu, long range forces on the cpu
- Each node gets either a region or a particle set
- If region is constant can get load imbalance
- If particle count is constant then dividing the short and long range forces is hard

Hybridise

- Need smaller domains on each node for load balancing

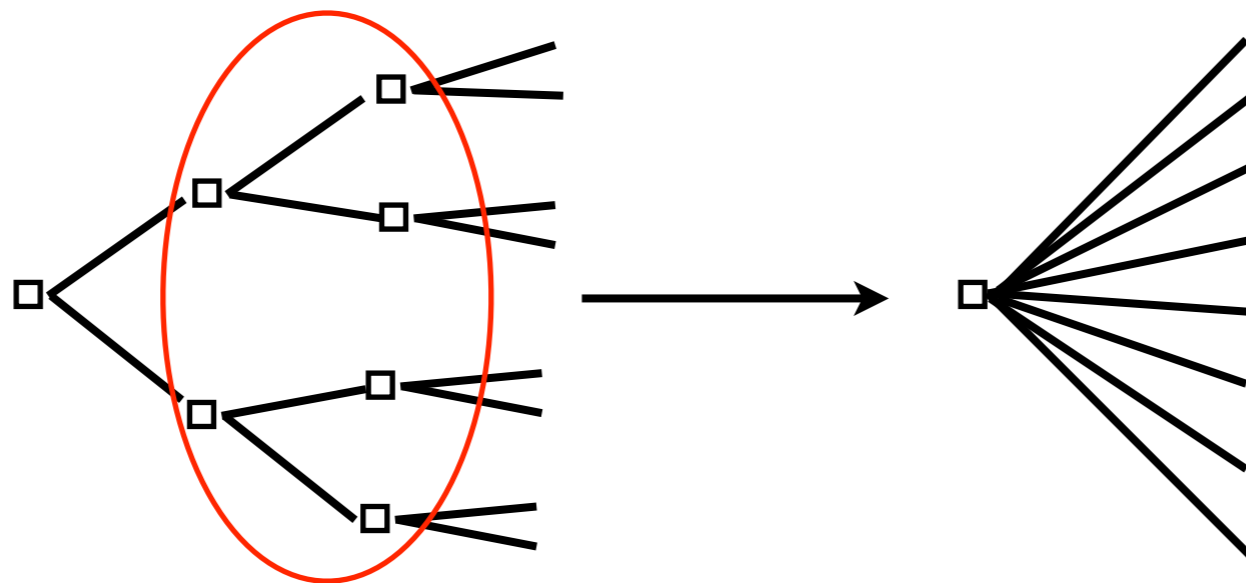


Tree on a GPU?

- Can we go further with subdivision - create a tree solver on the GPU?
- Remember we want each GPU core doing the same thing

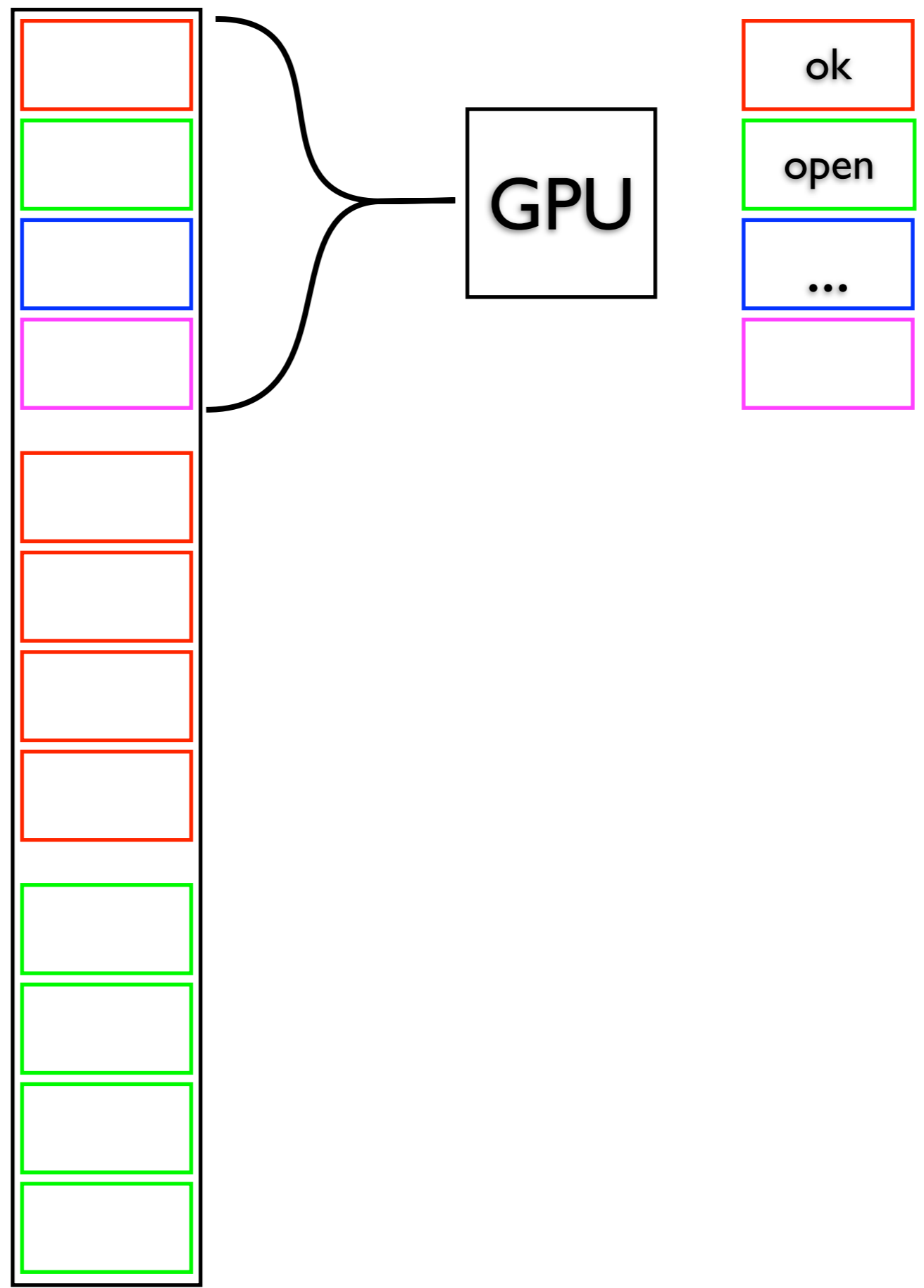
Tree on a GPU?

- Open multiple levels at once
- Each node effectively has more children - feed all these to the GPU cores



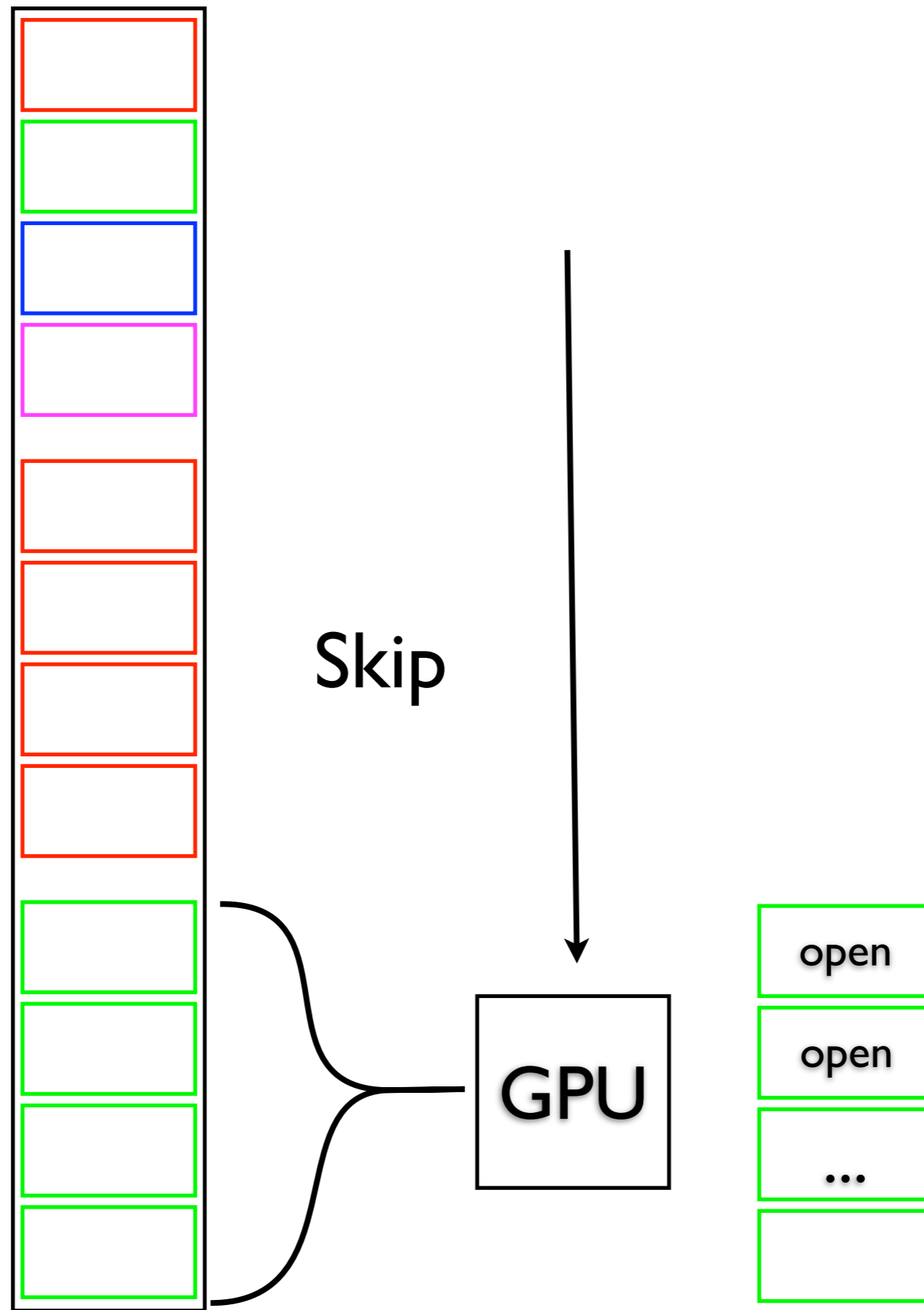
Optimising for Speed

- Main bottleneck is memory access
- Need to enforce data locality
- Don't load unneeded data to the GPU
- Registers are at a premium
- Need a smart data layout



Solving The N-body Problem Using GPUs

Scott Wales March 2010



Solving The N-body Problem Using GPUs

Scott Wales March 2010

Optimising for Speed

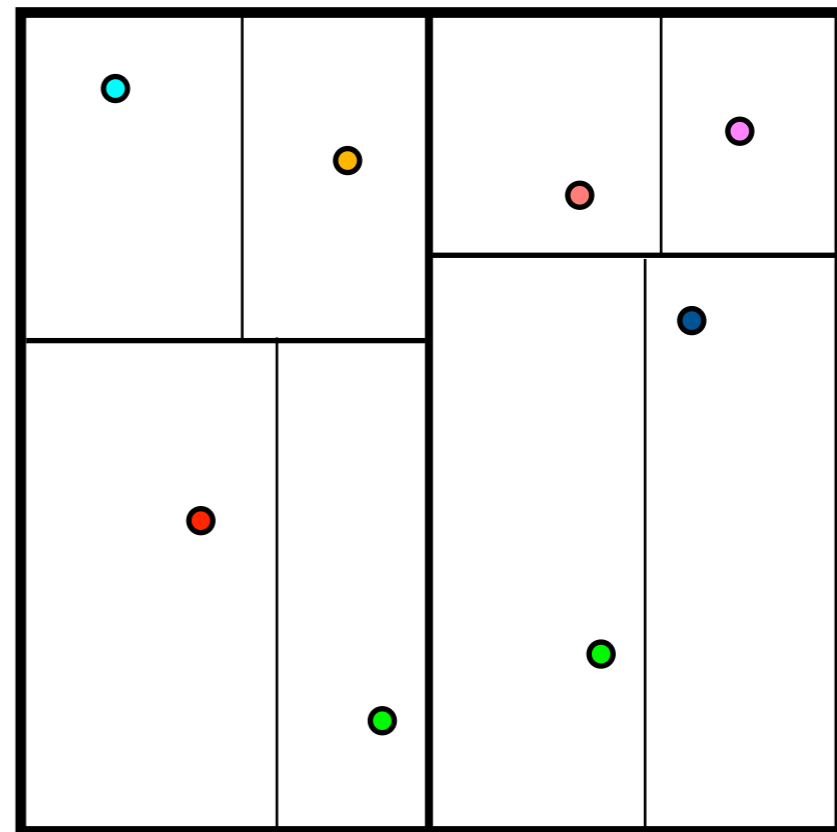
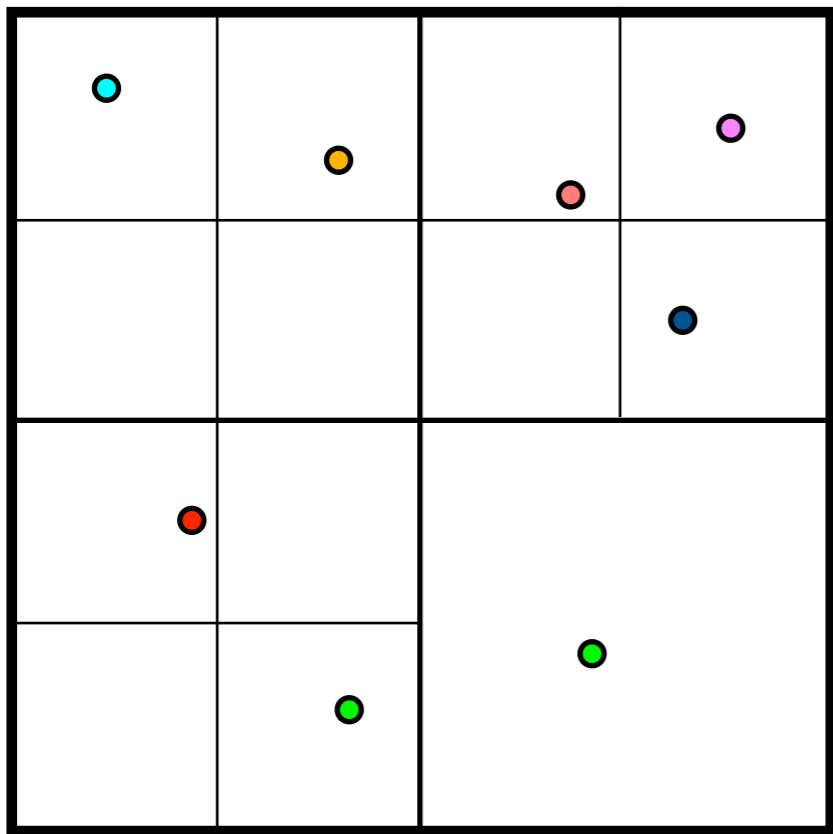
- Keeps each processor group occupied
- Same operation on each processor
- Long range forces can be calculated by the CPU while the GPU is in use

Memory Layout

- For force calculation, inputs are the tree moments, particle positions and masses, outputs are the short range forces
- Native data type for vectors is 4 floats - can hold positions and mass for each particle

Memory Layout

- Easiest if each node has the same number of children
- Divide the regions using a kD-tree instead of equal sizes - gives good data locality
- First levels won't have much for the GPU to do- can do this stage for multiple particles at once



Solving The N-body Problem Using GPUs

Scott Wales March 2010

Optimising for Accuracy

- Able to vary the number of tree nodes skipped over, the tree opening criteria, buffer regions, etc.
- Limited to 4 byte floats on the GPUs
- Possible to trade memory for accuracy using Kahan summation - 2 floats can emulate 1 double

Conclusions

- Hybrid Tree-Particle-Mesh method using GPUs
- Should give substantial improvements over a naive $O(n^2)$ algorithm