ChaNGa: Design Issues in High Performance Cosmology

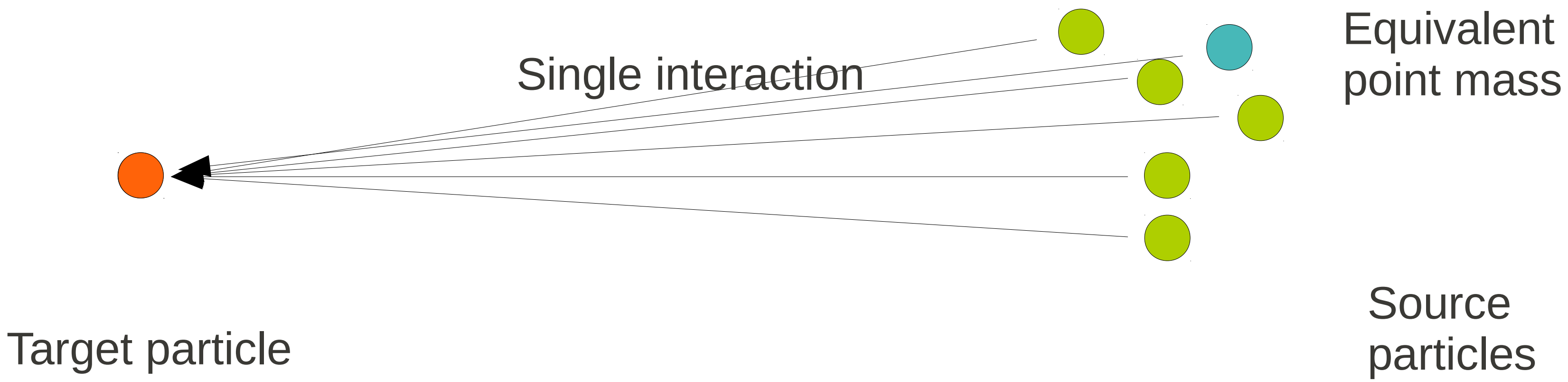
Pritish Jetley
Parallel Programming Laboratory

Overview

- Why Barnes-Hut?
- Domain decomposition
- Tree construction
- Tree traversal
  - Overlapping remote and local work
  - Remote data caching
  - Prefetching remote data
  - Increasing local work
  - Efficient sequential traversal
- Load balancing
- Multistepping

Why Barnes-Hut?

- Gravity is a long-range force
  - Every particle interacts with every other
- Do not need $N(N-1)/2$ interactions
- Groups of *distant* particles ≈ point masses
- $O(N \lg N)$ interactions



Single interaction

Equivalent point mass

Source particles

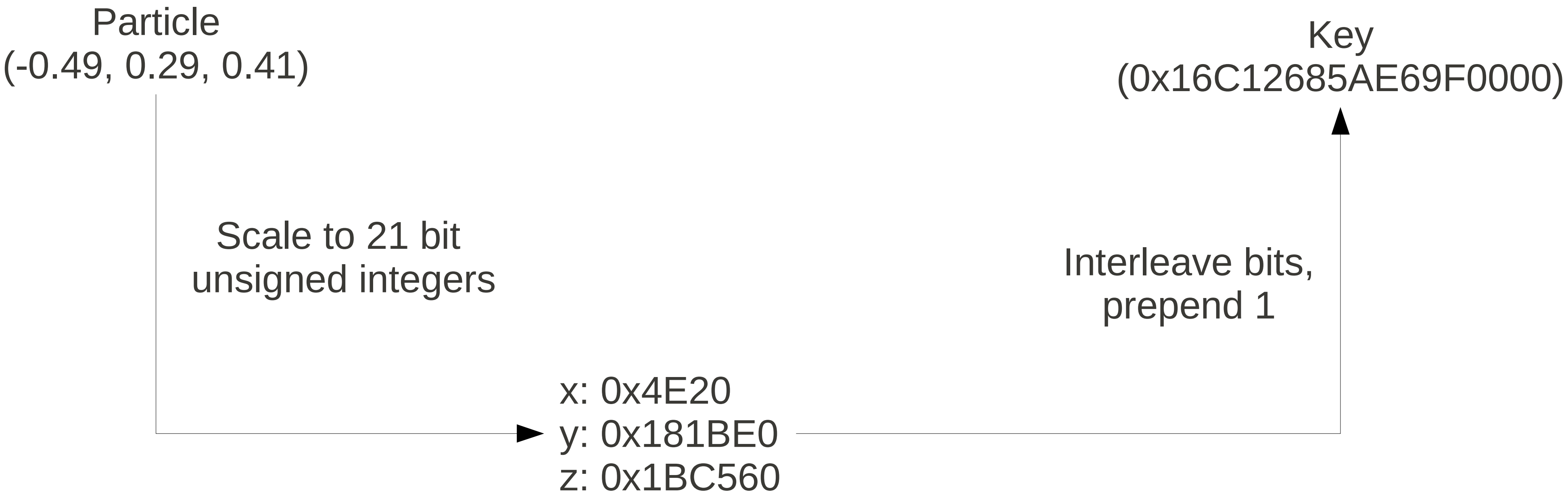
Target particle

Parallel Barnes-Hut: Decomposition

- Distribute particles among objects
- To lower communication costs:
    - Keep particles that are close to each other on the same object
    - Make spatial partitions regularly shaped
- Balance number of particles per partition

Decomposition strategies

- SFC: Linearize particle coordinates
  - Convert floats/doubles to integers
  - Interleave bits of integers



Particle
(-0.49, 0.29, 0.41)

Scale to 21 bit
unsigned integers

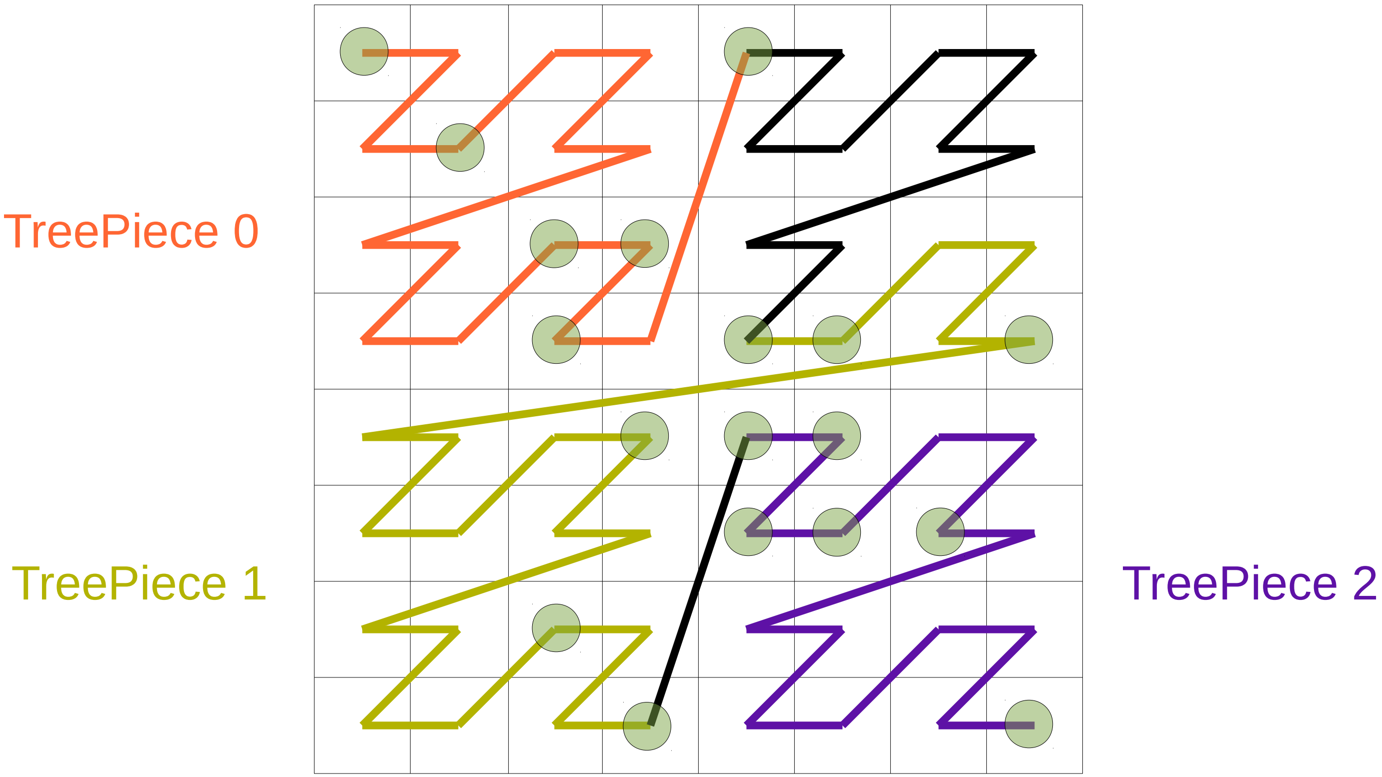
x: 0x4E20
y: 0x181BE0
z: 0x1BC560

Interleave bits,
prepend 1

Key
(0x16C12685AE69F0000)

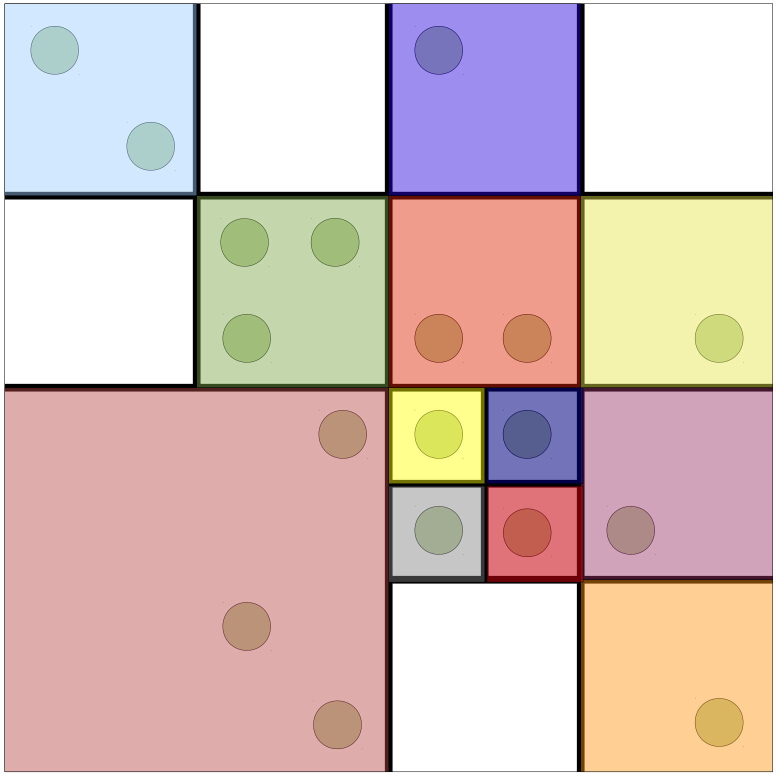
SFC

- Interleaving leads to jagged line of particles
- Line is split among objects (*TreePieces*)



Oct

- Recursively divide partition into quadrants if more than τ particles within it
- Iterative histogramming of particle counts

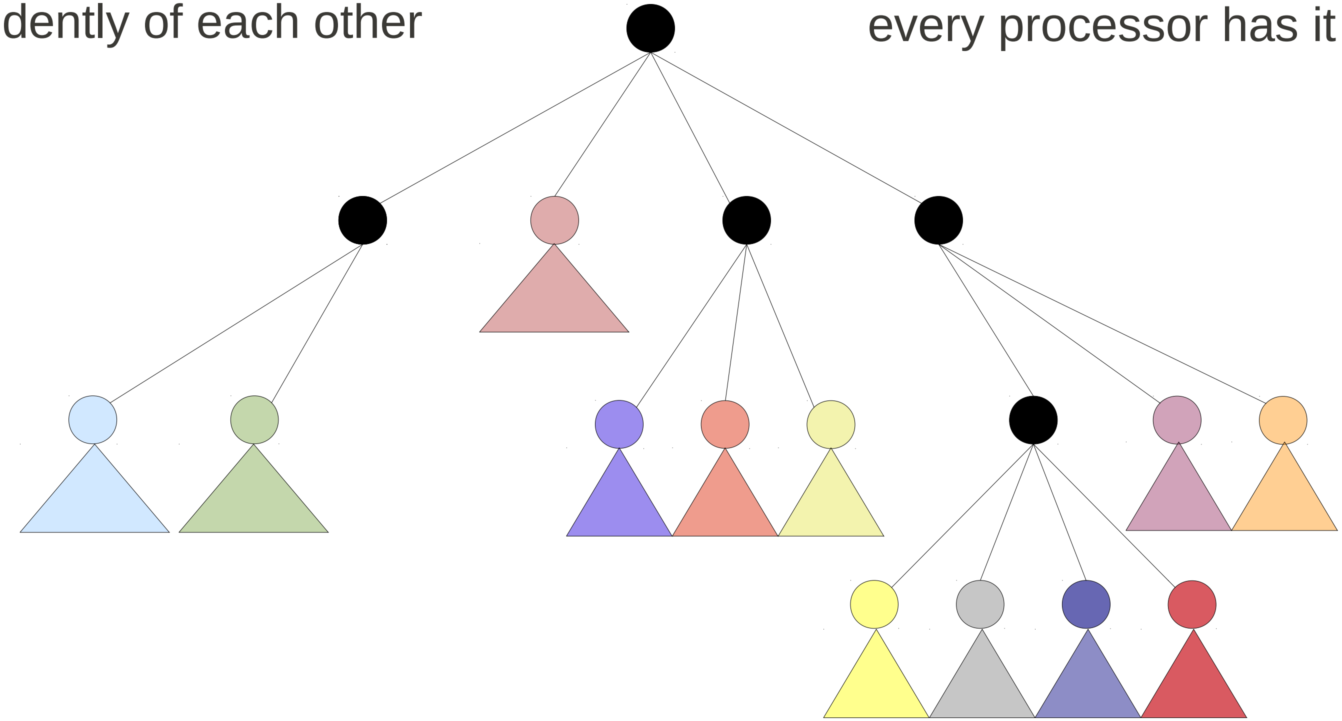


$\tau = 3$

Tree construction

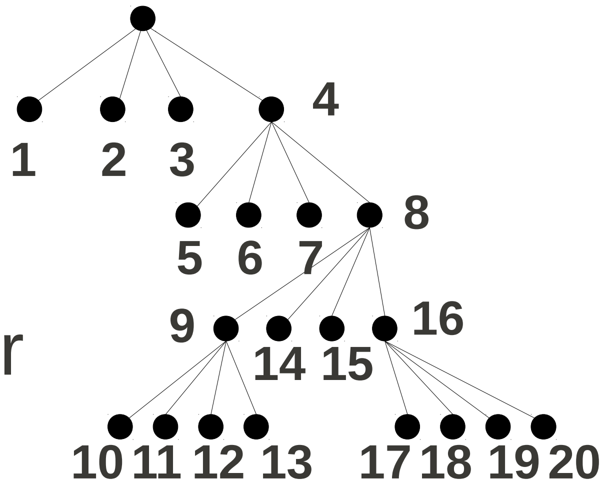
TreePieces construct trees beneath themselves independently of each other

Multipole moment information is passed up the tree so that every processor has it



Tree construction issues

- Must distribute TreePieces evenly across processors
- Particles stored as structures of arrays
  - (Possibly) more cache friendly
  - Easier to vectorize accessing code
- Tree data structure layout?
  - `new` for each node - BAD!
  - Better: allocate all children together
  - Better still: allocate in a DFS manner

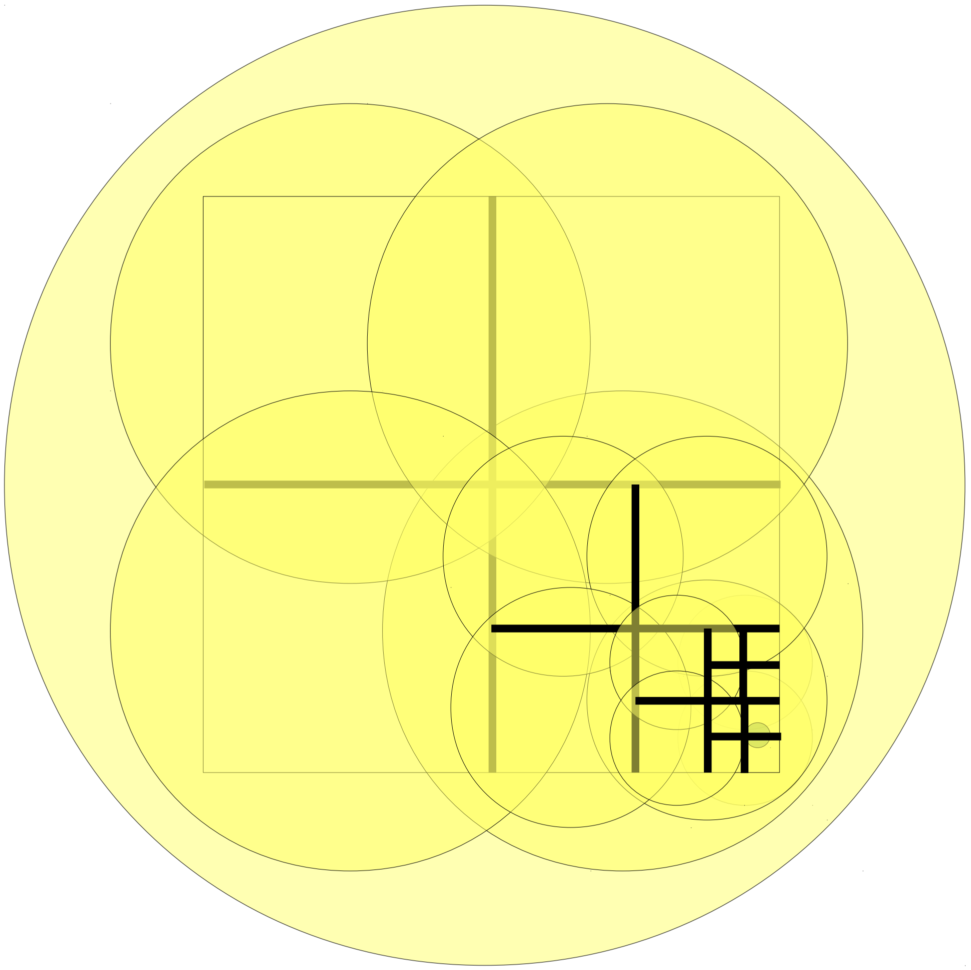


Tree traversal

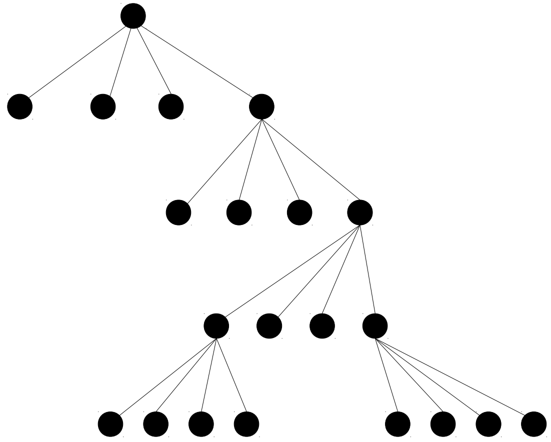
- A TreePiece performs **depth-first traversal** of tree for *each bucket* of particles
- For each node encountered,
  - Is node far enough?
    - Compute forces on bucket due to node
    - Pop node from stack
  - Node too close?
    - Push next child onto stack

Illustration

Yellow circles
Represent
Opening criterion checks



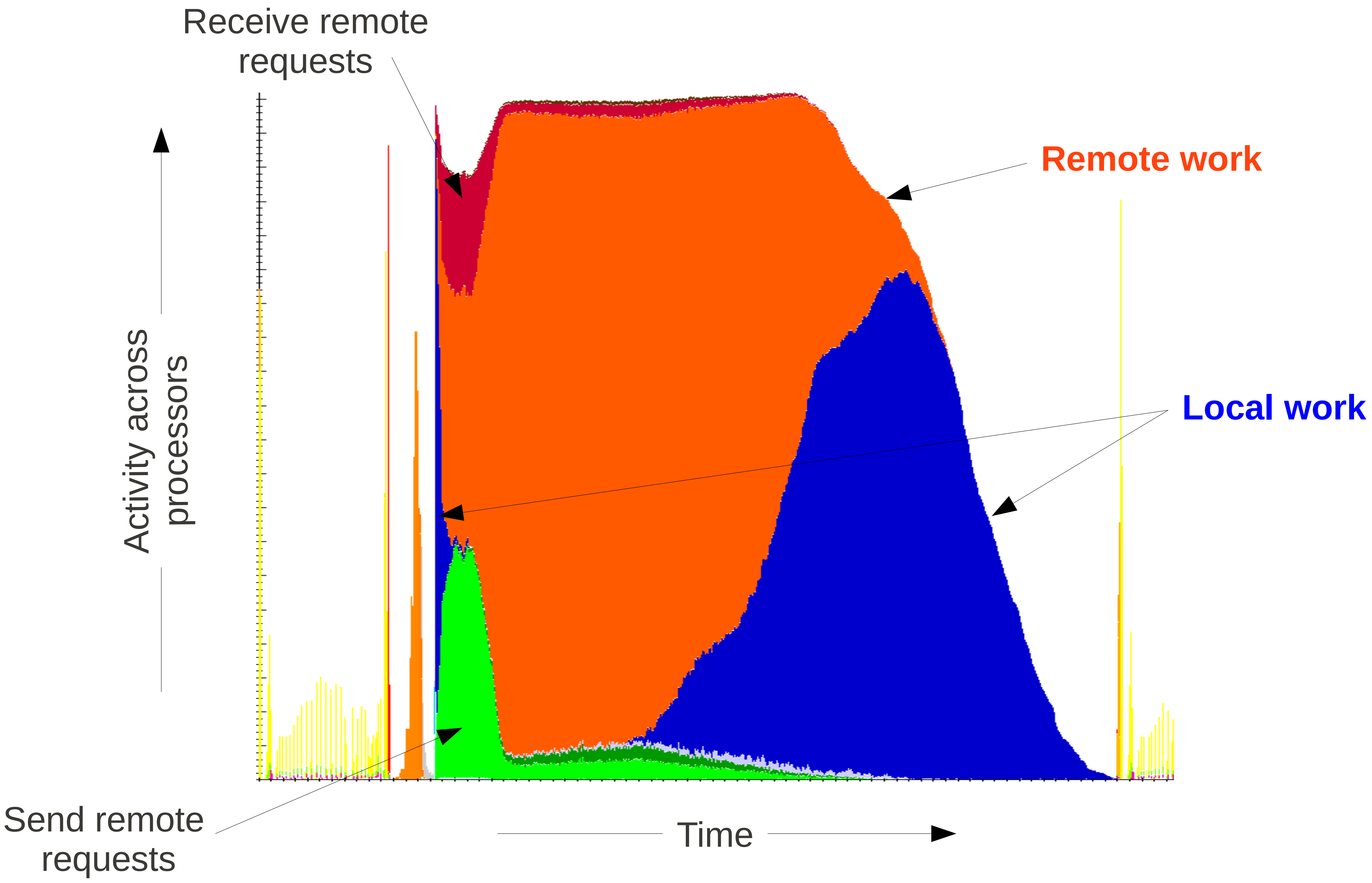
=



Tree traversal

- Cannot have entire tree on every processor
  - Local nodes
  - Remote nodes
- Remote nodes must be requested from other TreePieces
  - Generate communication
- Give **high priority** to remote work
  - Do local work when waiting for remote nodes to arrive: **overlap**

Overlapping remote and local work



Receive remote requests

Remote work

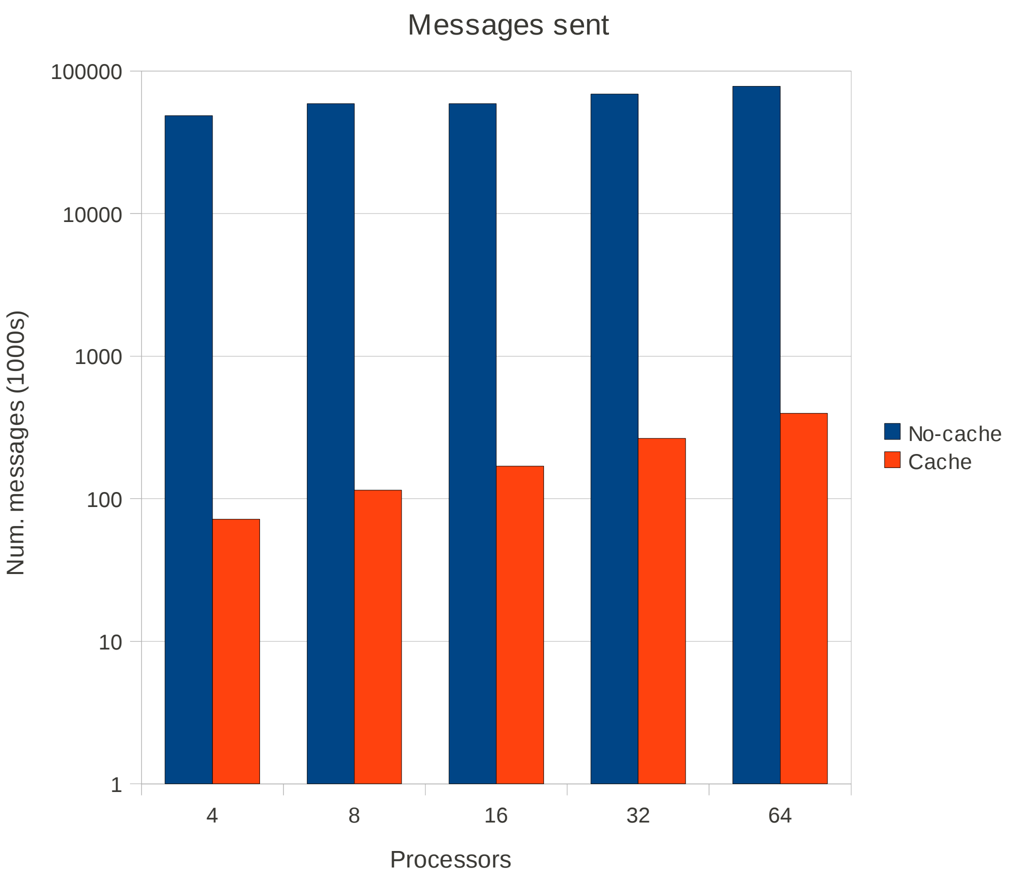
Local work

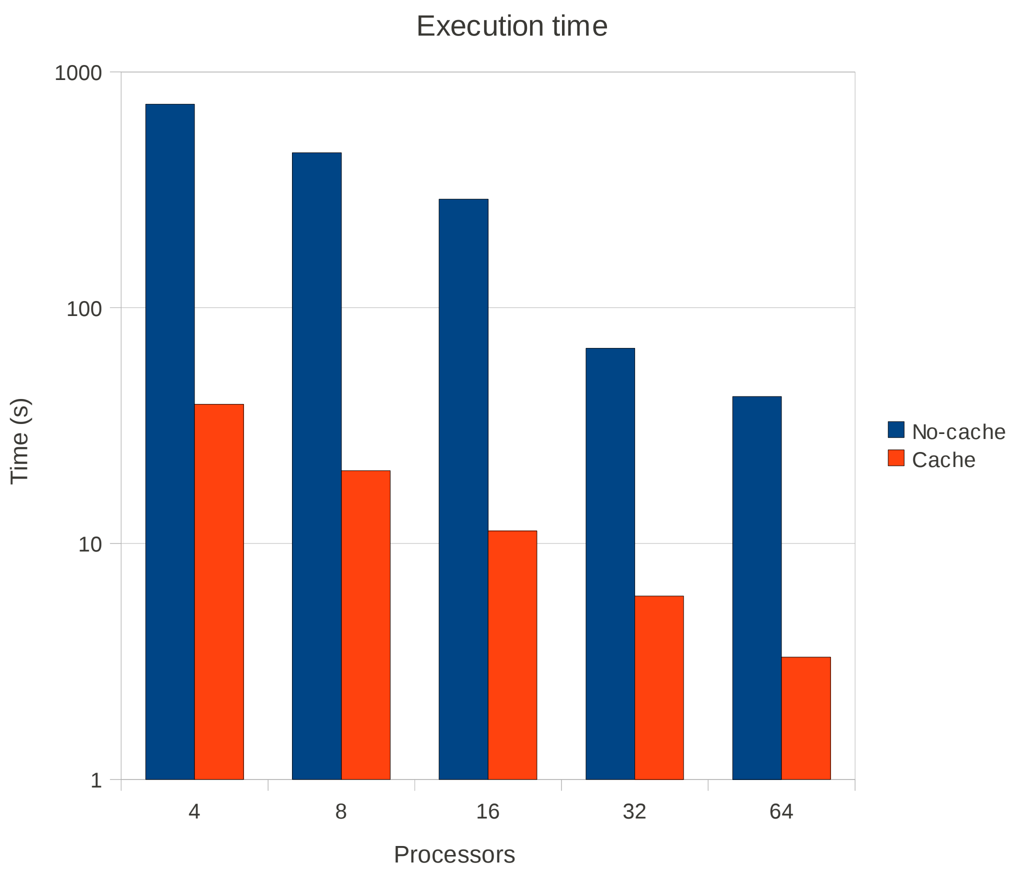
Send remote requests

Remote data caching reduces communication

- Reuse requested data to reduce number of requests
- **Cache requested remote data** on processor
  - Data requested by one TreePiece used by others
  - Fewer messages
  - Less overhead for processing remote data requests
- Optimal cache line size (depth of tree beneath requested node)
  - About 2 for Octrees

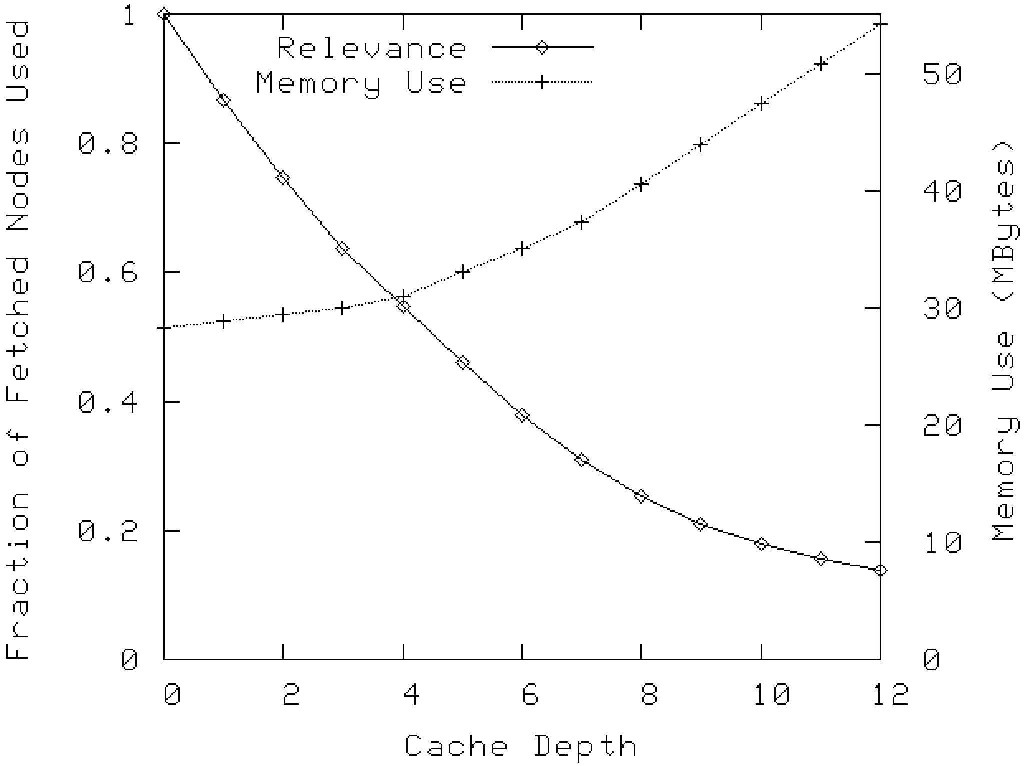
Remote data caching

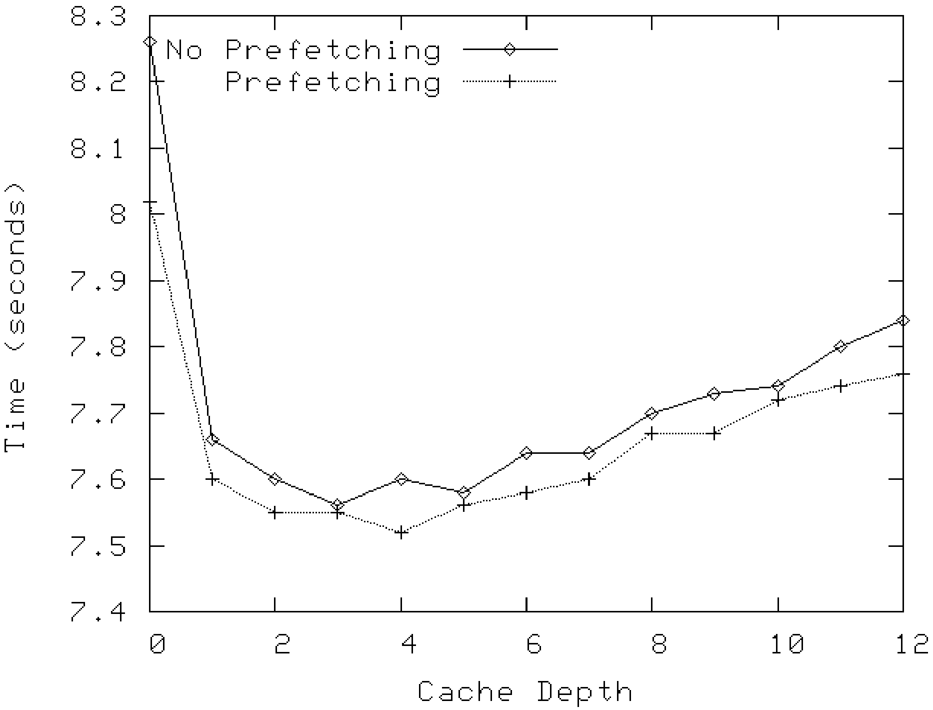




Remote data prefetching

- Estimate remote data requirements of TreePieces, prefetch before traversal
  - Reduces latency of node access during traversal





Increasing local work

- Division of tree into TreePieces reduces the amount of local work per piece
- Combine TreePieces in one processor to increase amount of local work
  - Without combination, 16% local work per TreePiece
  - With combination, 58%

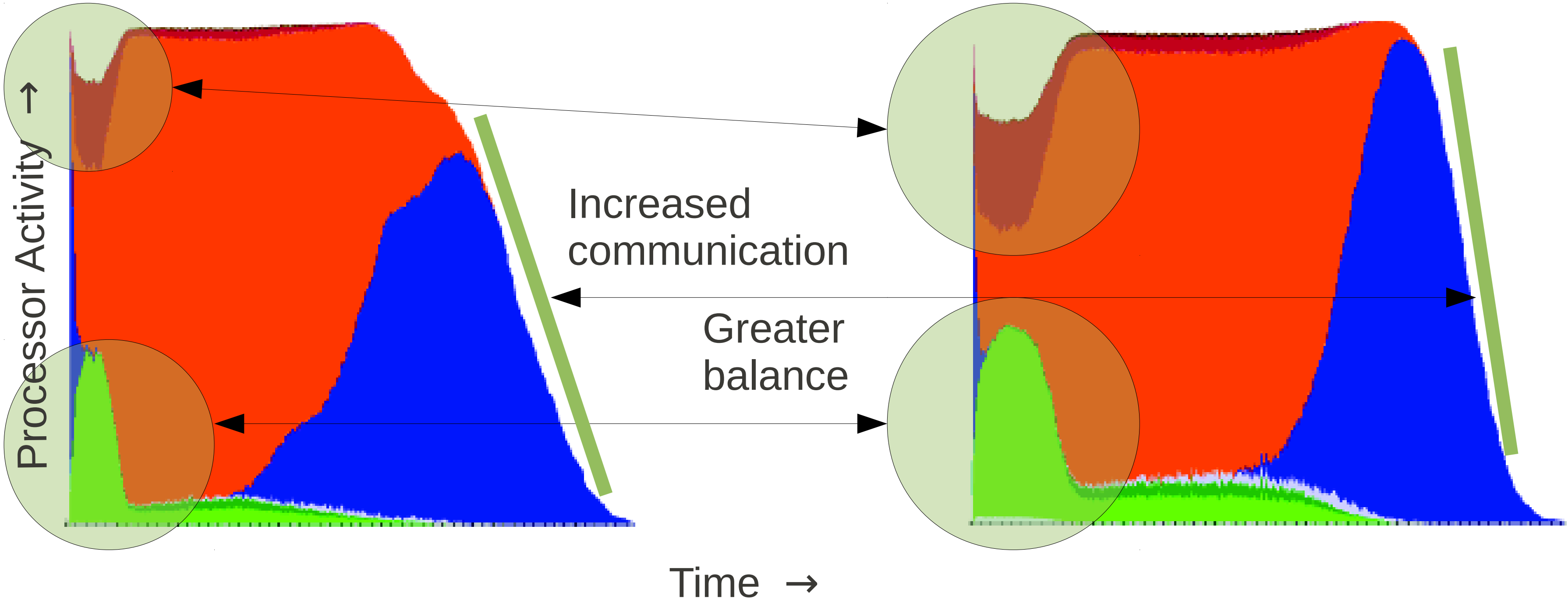
Algorithmic efficiency

- Normally, walk entire tree once for each bucket
- However, proximal buckets have similar interactions with the rest of the universe
- Share lists between buckets as far as possible
  - Check distance between
    - Remote tree node
    - Local ancestor of buckets (instead of buckets)
- Improvements of 7-10% over normal traversal

Load balancing

- Density variations in input data create load imbalance
- Load balancing must account for computation as well as communication

Balancing Load to Improve Performance



Increased communication

Greater balance

🟦🟧 Computation

🟩🟥 Communication

LB algorithms must consider both computation and communication

Multistepping

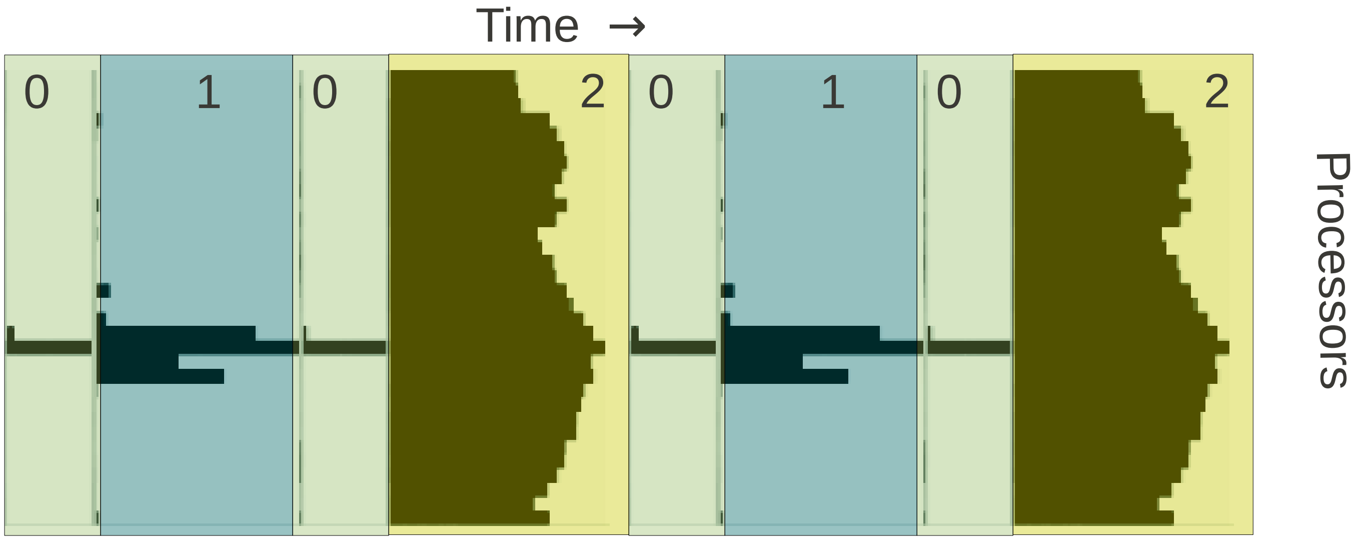
- Group particles into *rungs*
  - Faster rung → more speed
  - Different rungs active at different times
- Update slower rung particles **less frequently**
- Less computation done than singlestepping

Computation split into *phases*

0: rung 0

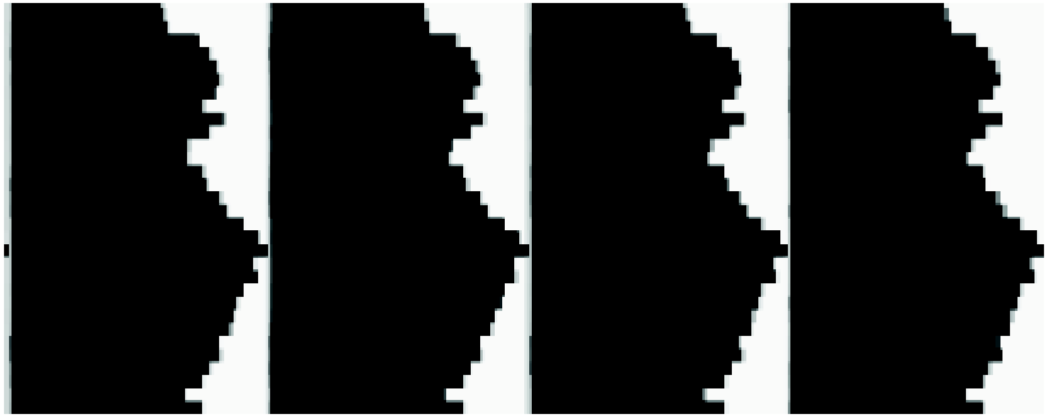
1: rungs 0,1

2: rungs 0,1,2

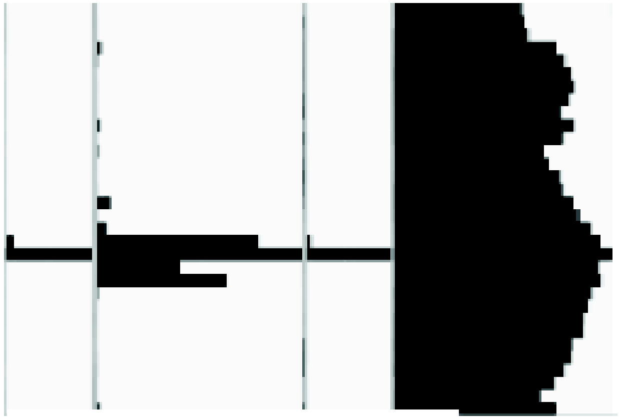


Load imbalance with multistepping

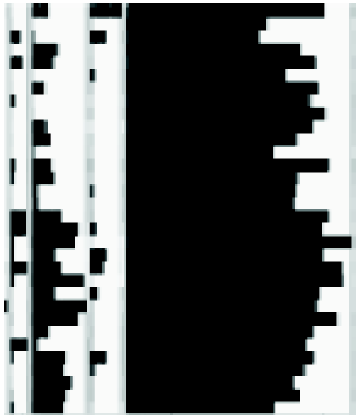
- Dwarf dataset
- 32 BG/L processors
- Different timestepping schemes



Singlestepped
(613 s)



Multistepped
(429 s)



Multistepped
with
load balancing
(228 s)

Thank you

Questions?