CMSC 714 paper reading round 2

A Parallel Hashed Oct-Tree N-Body Algorithm

Jing Xie

April 22, 2021

N-body algorithm basic idea

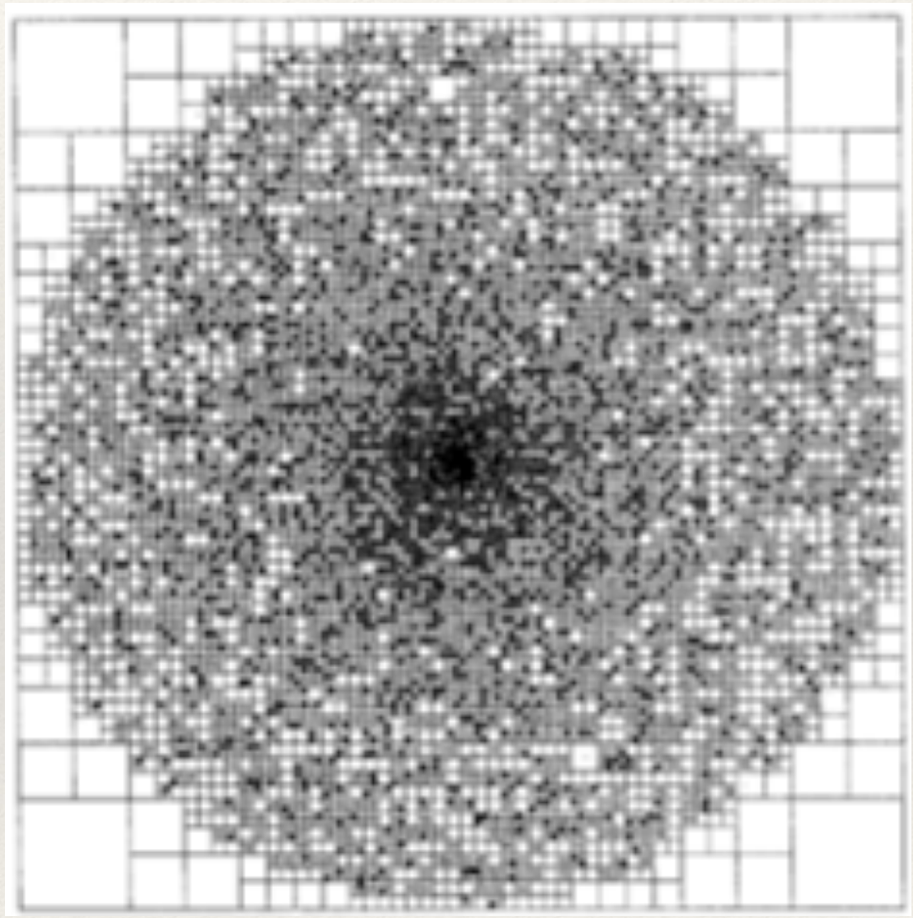
- Partition an arbitrary collections of bodies in a manner where series approximation can be applied to pieces.
- Maintain sufficient accuracy in the force on each particle.
- A system of N bodies is represented hierarchically.
- Spatial tree data structure.
- Cells.
- Expansion has limited domain of convergence.
  - Multipole acceptance criterion (MAC) make a choice of cells.

Problems

- N-body simulation is a fundamental tool to study complex physical systems.
  - MAC described above is problematic because the parallel algorithm requires determination of locality essential data before the tree traversal begins.
  - Data dependent MAC is difficult to pre-determine which non-local cells are required in advance of the traversal stage.

Multipole Methods

- Appel: binary tree data structure whose leaves are bodies, internal nodes are spherical cells.
- Barnes-Hut algorithm: regular, hierarchical cubical subdivision of space.
- Fast multipole method (FMM): high order multipole expansions and interacts fixed sets of cells.

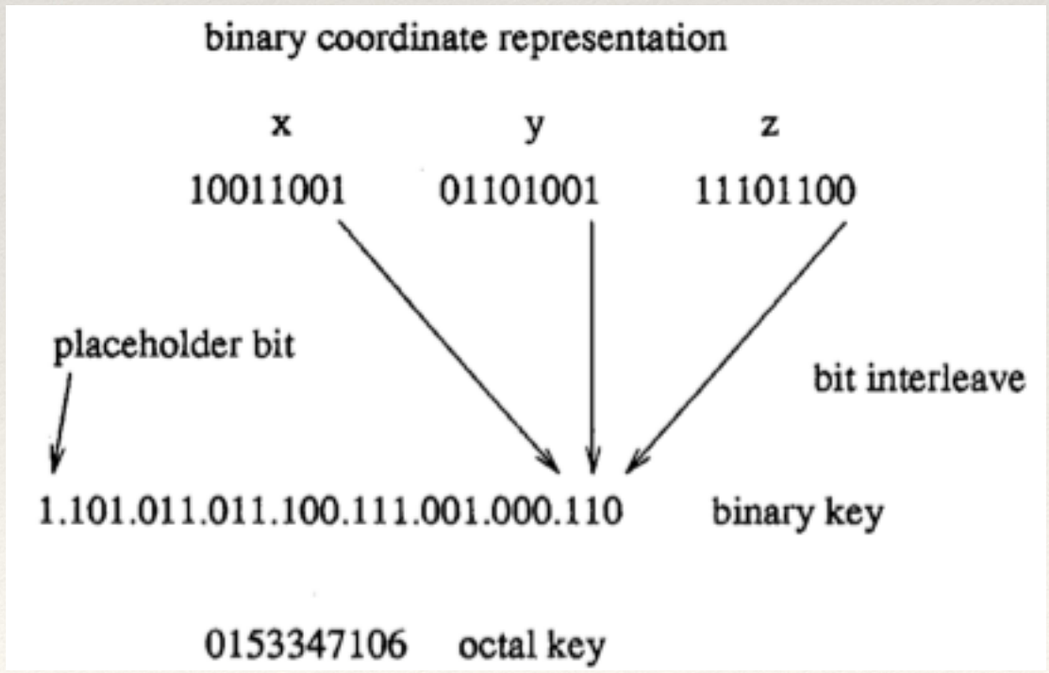


Key construction and hashing functions

- Identifying each possible cell with a key.
- Produce the keys of daughter or parent cells by simple bit arithmetic.
- Given a key, data can be quickly retrieved.

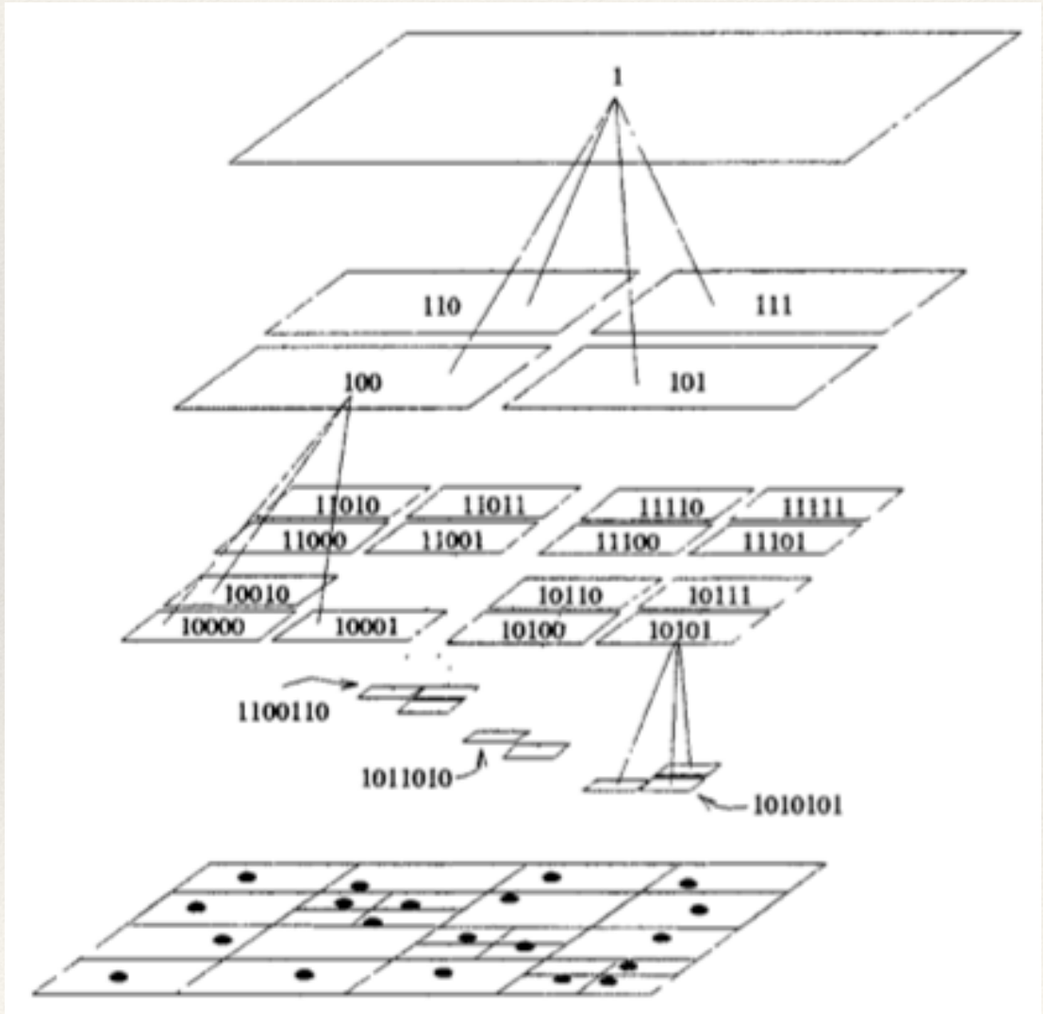
Key construction

- A key is the result of a map of *d* floating point numbers into a single set of bits.
- Translate the floating point numbers into int
- Interleave the bits of the *d* integers into a single key.



Key construction

- A quad-tree with the binary key coordinates of the nodes.



hashing functions

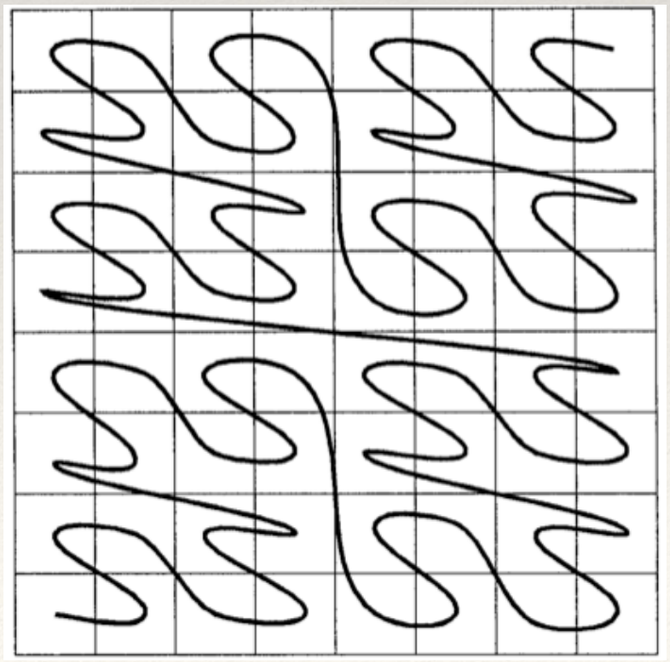
- A hash table is used to map the key to the memory location holding this data.
- Hash function:
  - *k*-bit key --> *h*-bit hash address
  - AND the key with the bit-mask 2^h -1 to select least significant *h* bits.
- A linked list resolves the collisions in hash table.
- Key space is convenient for tree traversals
- More advantages

Tree construction

- Simple way: Particle is loaded into the tree by starting at root and traverse the partially constructed tree.
- Faster method: Sort the body keys, consider the body list in order. Start traversing at the location of the last node as bodies are inserted into the tree.

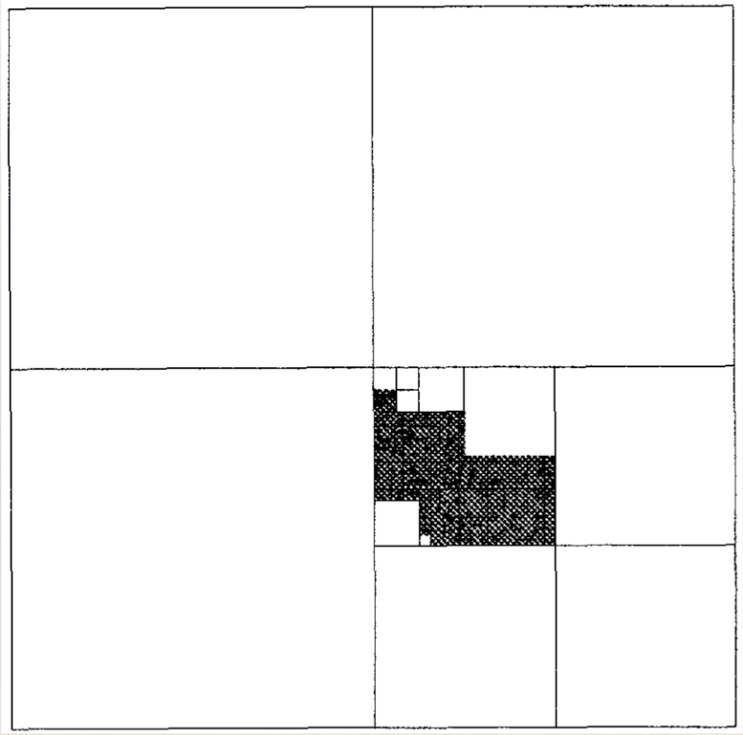
Parallel data decomposition

- Cut the 1-d list of sorted body key ordinates into N_p equal pieces, weighted by the amount of work of each body.
- Each body's work is readily approximated by counting the number of interactions that the body is involved previously.



Parallel tree construction

- Construct a tree made up of local bodies.
- Send a copy of each boundary body to the adjacent processor. Copies of branch nodes from each processor are shared among all.
- These branch cells are globally communicated. All processors can fill in the missing part of the tree. The address of processor which owns each branch cell is passed to the destination processor. Hcell is marked with its origin.
- Traversal routine determines which processor to request data from.



Tree traversal

```
Traverse(Key_t key, int (*MAC)(hcell *),
         void (*postf)(hcell *)) {
    hcell *pp;
    unsigned int child;

    if ((pp=Find(key)) && MAC(pp)) return;
    key = KeyLshift(key, NDIM);
    for (child = 0; child < (1<<NDIM); child++)
      Traverse(KeyOrInt(key, child), MAC, postf);
    postf(pp);
```

```
ListTraverse((*MAC)(hcell *))
{
  copy root to walk_list;
  while (!Empty(walk_list)) {
    for (each item on walk_list) {
      for (each daughter of item) {
        if (MAC(daughter))
          copy daughter to interact_list;
        else
          copy daughter to output_walk_list;
      }
    }
    walk_list = output_walk_list;
  }
}
```

Performance

computation stage	*time* (sec)
Domain Decomposition	7
Tree Build	7
Tree Traversal	33
Data Communication During Traversal	6
Force Evaluation	54
Load Imbalance	7
Total (5.8 Gflops)	114

computation stage	*time* (sec)
Domain Decomposition	19
Tree Build	10
Tree Traversal	55
Data Communication during traversal	4
Force Evaluation	60
Load Imbalance	12
Total (4.9 Gflops)	160

Summary

- An efficient adaptive N-body method
  - Compute the forces on an arbitrary distribution of bodies in O(NlogN) time scale.
    - Accuracy is analytically bounded and adjustable
- Identify each possible cell with a key to access data more efficiently.
- Parallel Performance is evaluated on 512 processor Intel Touchstone Delta system.

Thanks