THE UNIVERSITY OF CHICAGO

Petascale Multiscale Simulations of Biomolecular Systems

John Grime
Voth Group
Argonne National Laboratory / University of Chicago

CENTER for MULTISCALE THEORY and SIMULATION
NSF CENTER for CHEMICAL INNOVATION

THE UNIVERSITY OF CHICAGO

About me

- Background: experimental guy in grad school (LSCM, drug delivery) – I seem to have become a theoretical chemist. I'm as confused as anyone regarding how *that* happened.

- **Argonne National Labs / University of Chicago**

- Talk is about a side project I've been working on amongst my usual day-to-day activities; it's at an early stage!

- **This is a via a PRAC sub-award**

CENTER for MULTISCALE THEORY and SIMULATION
NSF CENTER for CHEMICAL INNOVATION

Molecular dynamics

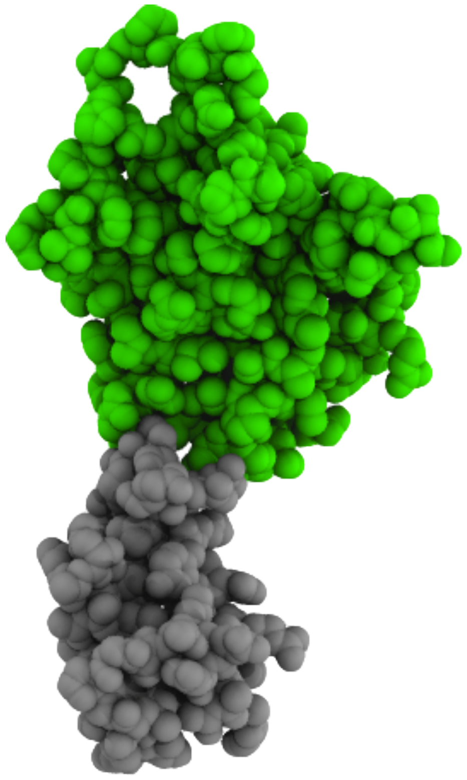
MD (classical) numerically integrates molecular equations of motion:

$$\boldsymbol{F = ma}$$

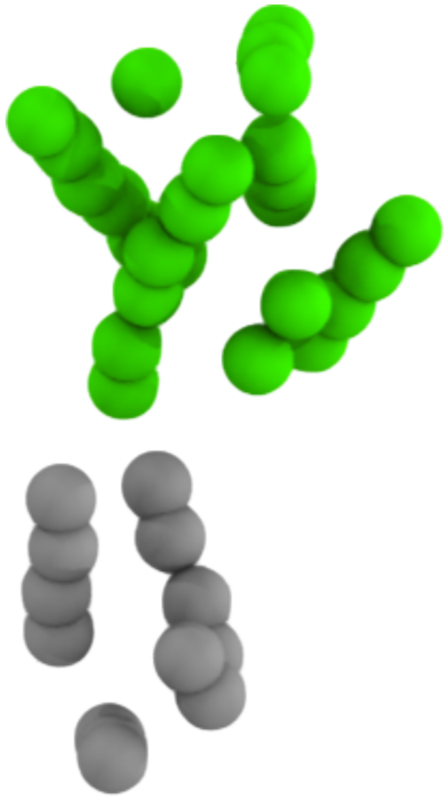
- One or more atoms, linked by springs
- Typically, also special springs to maintain structure (angles formed between 3 atoms etc)
- Interactions between molecules are distance dependent (get weaker with distance)
- **“True” atomic level-of-detail is computationally expensive!**

THE UNIVERSITY OF CHICAGO

“Coarse graining” (CG)



Make the model simpler, while maintaining the essential behaviours!



Atomic model

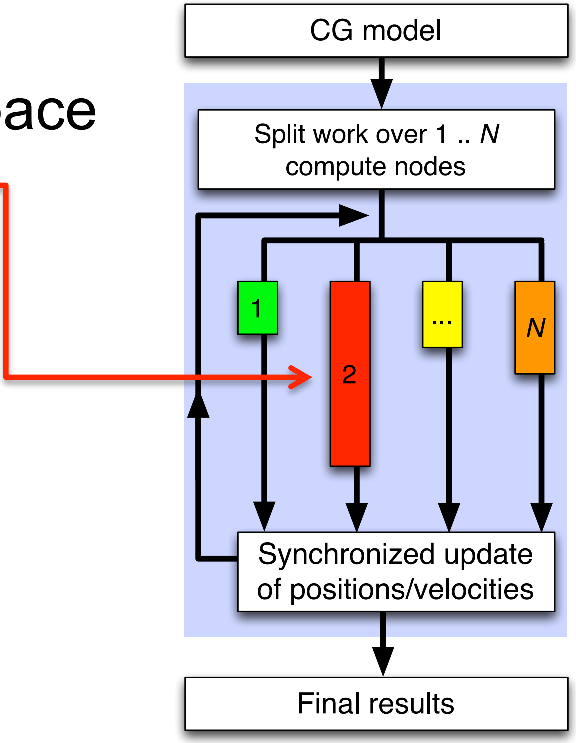
CG model

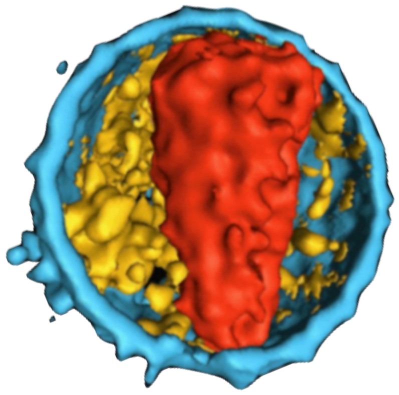
CENTER for MULTISCALE THEORY and SIMULATION
NSF CENTER for CHEMICAL INNOVATION

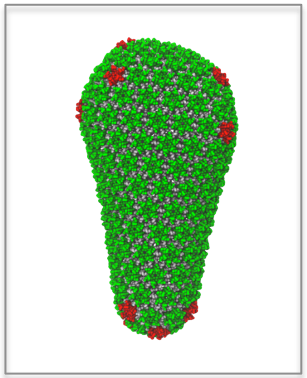
CG-MD problems

Implicit solvent for large scale ultra-CG: introduces large, dynamic areas of low particle density:

- *Load balancing*: MD simulation proceeds at pace of **slowest node**


CG model
Split work over 1 .. *N* compute nodes
1
2
...
N
Synchronized update of positions/velocities
Final results


HIV capsid[1]


Ultra-CG model

- *Memory requirements*: memory needed even for empty regions of simulation

[1]Tom Goddard 2009, UCSF

CG-MD problems

A new MD code for multiscale CG:

- **Dynamic sparse data** representations (removes memory barriers for very large systems)

- **Load balancing** via Hilbert space filling curves (better use of supercomputing resources)

End goal: enable highly dynamic CG-MD simulations at a **cellular scale**!

CENTER for MULTISCALE THEORY and SIMULATION
NSF CENTER for CHEMICAL INNOVATION

THE UNIVERSITY OF CHICAGO

“Sparse” data

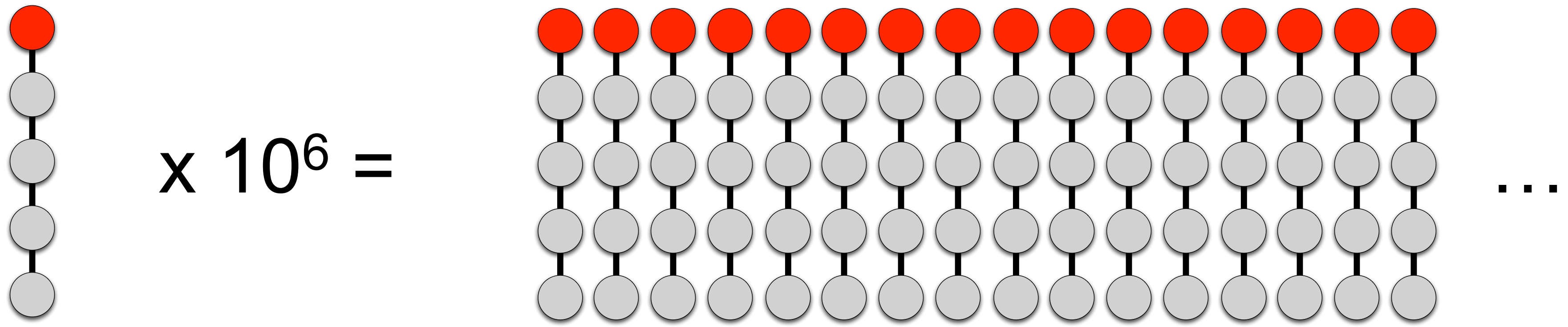
Two fundamentally different approaches vs standard MD:

- No global bonding topology – topology is local and *implicit*, allows significant dynamic runtime changes (add/remove/modify molecules etc)

- Reduced “link cell” memory requirements

CENTER for MULTISCALE THEORY and SIMULATION
NSF CENTER for CHEMICAL INNOVATION

"Sparse" data



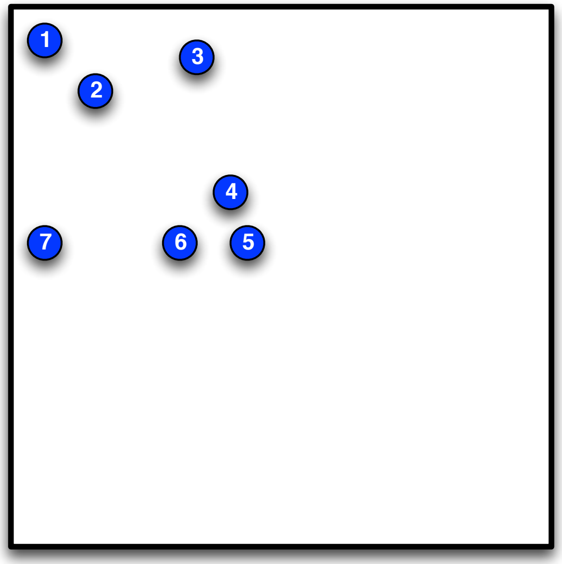
1M CG lipids = **~240 megabytes** of disk space in LAMMPS for bonding topology*: *but the same information is repeated for all lipids!*

New CG code needs **364 *bytes***. *Always*. One single "template" lipid structure, local topology calculated dynamically

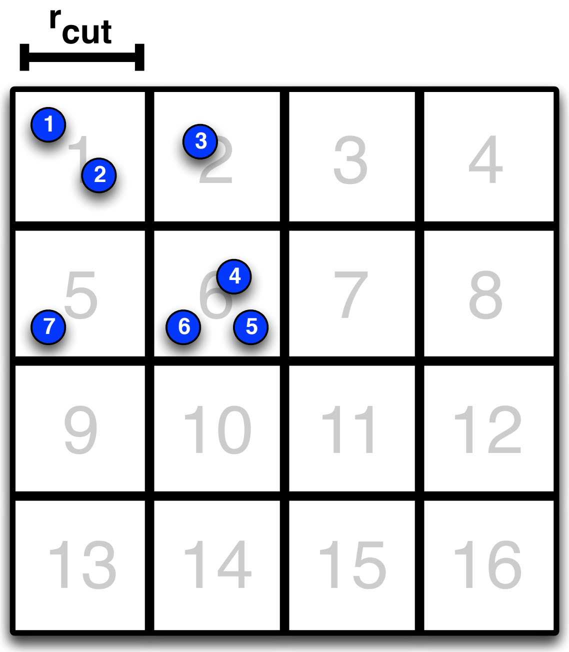
*4 bonds + 3 angles

THE UNIVERSITY OF CHICAGO

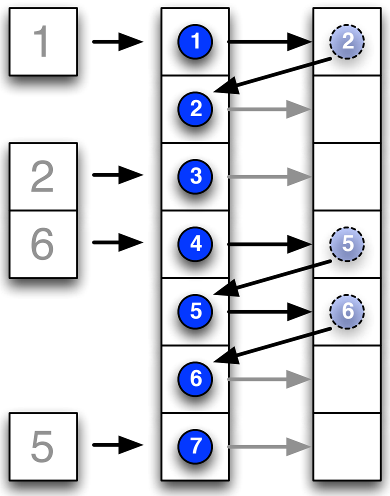
“Sparse” data



System



Divide into cells of size r_{cut}

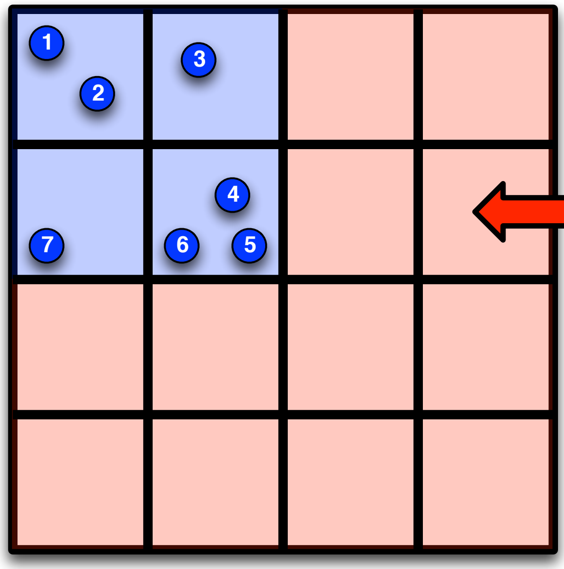


Linked list of atoms in each “link cell”

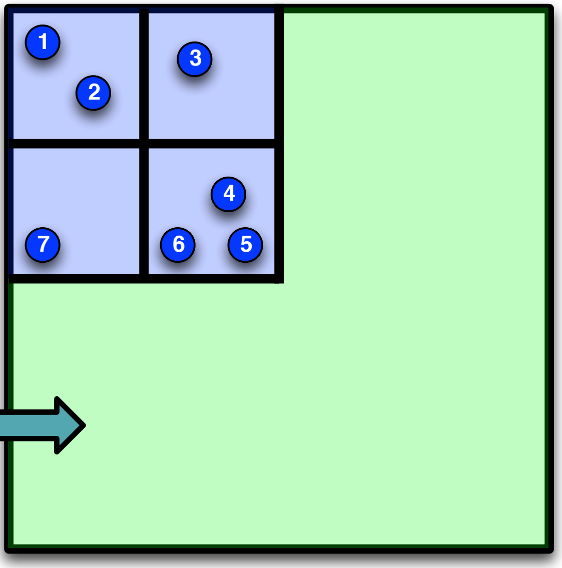
“Link cell” algorithm used behind the scenes in MD – fast calculation of nonbonded interactions, but …

CENTER for MULTISCALE THEORY and SIMULATION
NSF CENTER for CHEMICAL INNOVATION

“Sparse” data

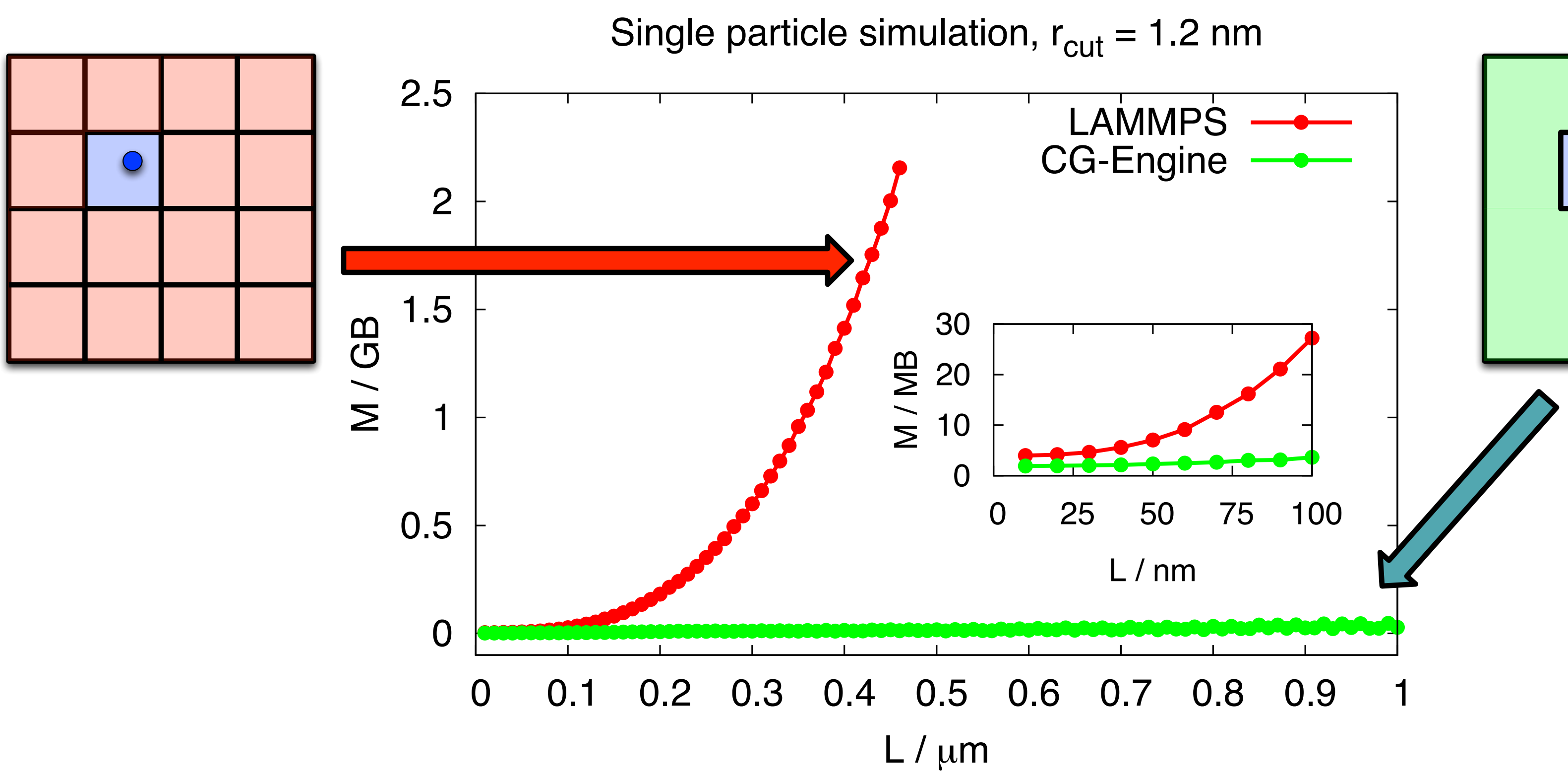


… conventional link cell algorithm: red cells require memory, even when empty!



CG-MD “sparse” link cells: green region requires no memory if empty!

“Sparse” data

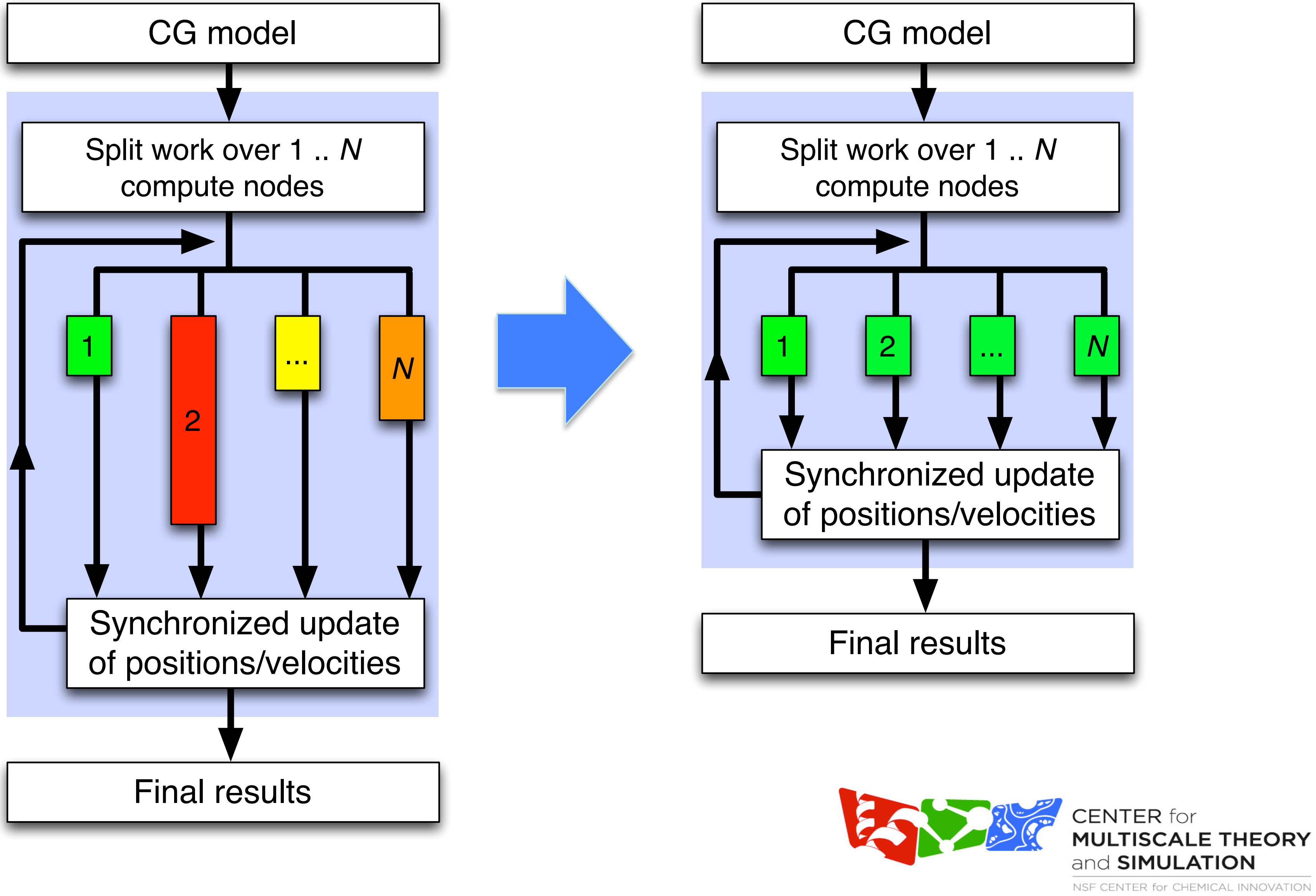


Trivial example: **single particle** in large volume uses *huge* amounts of memory – yet the simulation is basically empty!

New CG code is much better behaved.

THE UNIVERSITY OF CHICAGO

Load balancing

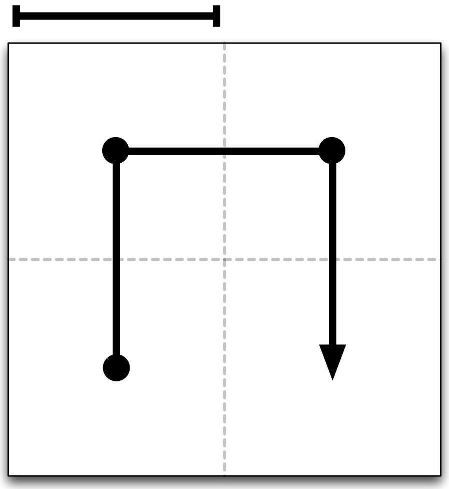

CG model
Split work over 1 .. *N* compute nodes
1
2
...
N
Synchronized update of positions/velocities
Final results
CG model
Split work over 1 .. *N* compute nodes
1
2
...
N
Synchronized update of positions/velocities
Final results

CENTER for MULTISCALE THEORY and SIMULATION
NSF CENTER for CHEMICAL INNOVATION

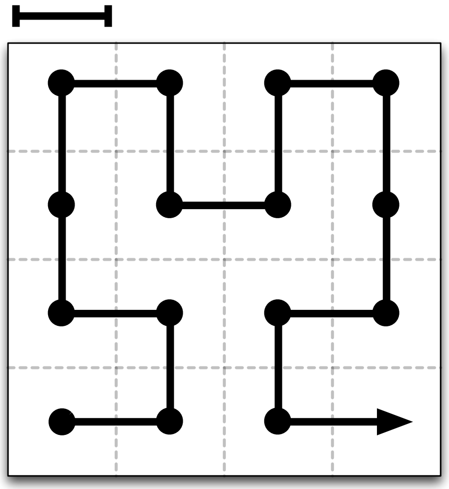
THE UNIVERSITY OF CHICAGO

Load balancing

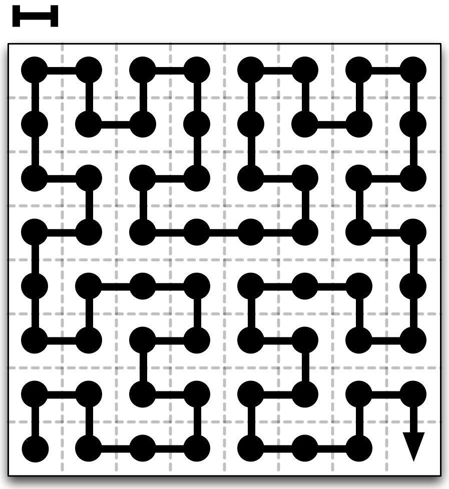
The space filling Hilbert curve



1st order



2nd order



3rd order

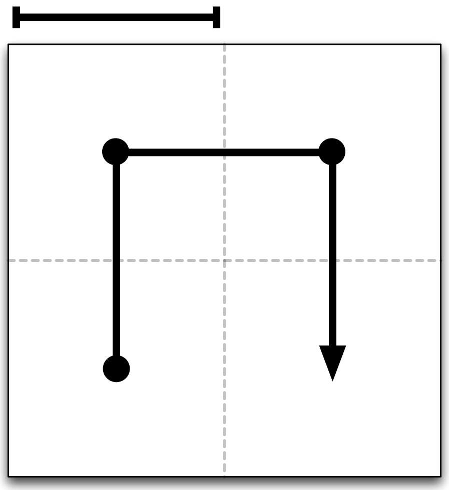
Useful properties:

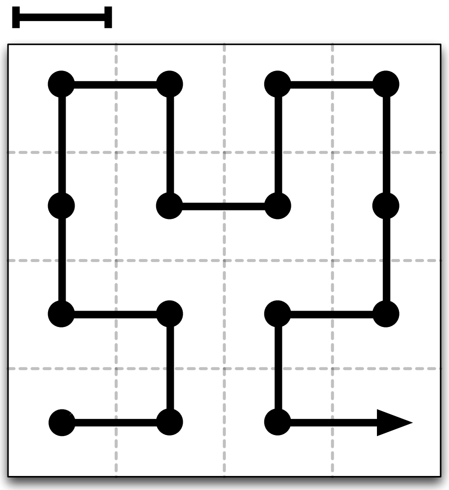
- **Space filling** (high resolution where needed)
- **Dynamic generation** (no precalculation)
- **Locality** (adjacent SFC "indices" also local in the original 2D/3D Cartesian space)

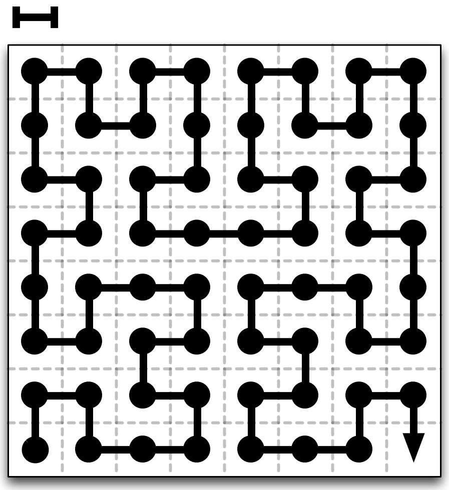
CENTER for MULTISCALE THEORY and SIMULATION
NSF CENTER for CHEMICAL INNOVATION

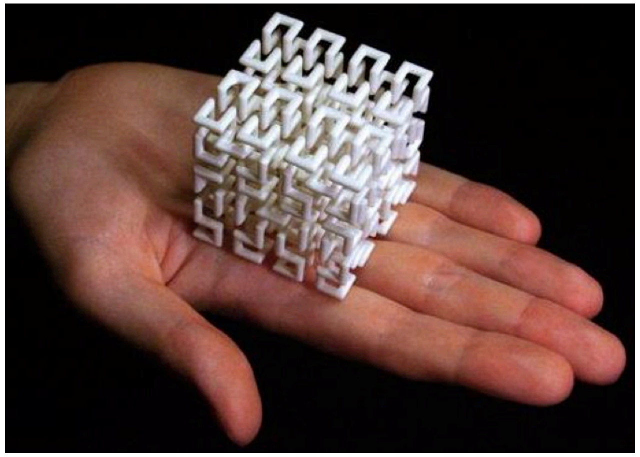
THE UNIVERSITY OF CHICAGO

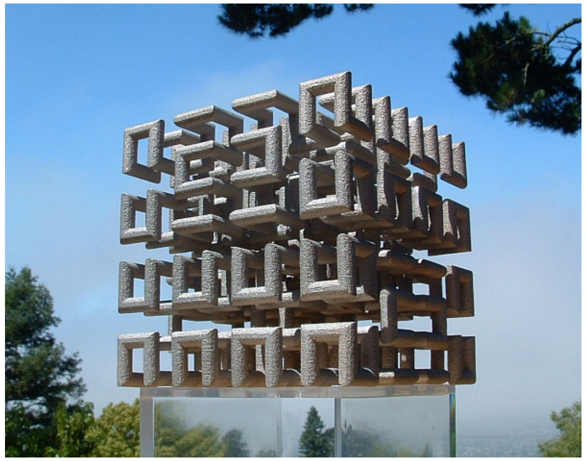
Load balancing









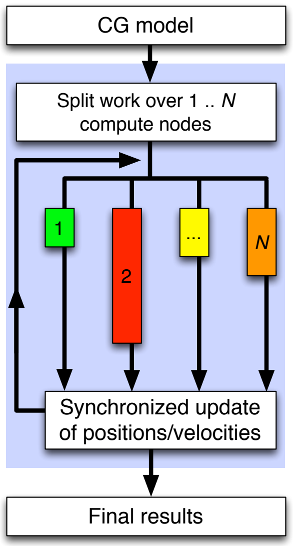


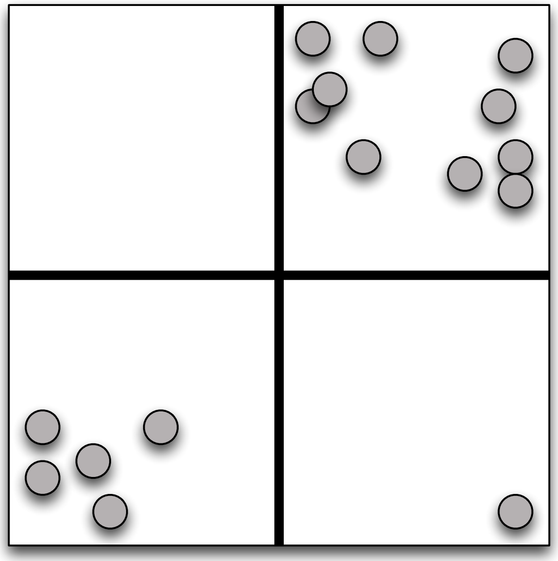
CENTER for MULTISCALE THEORY and SIMULATION

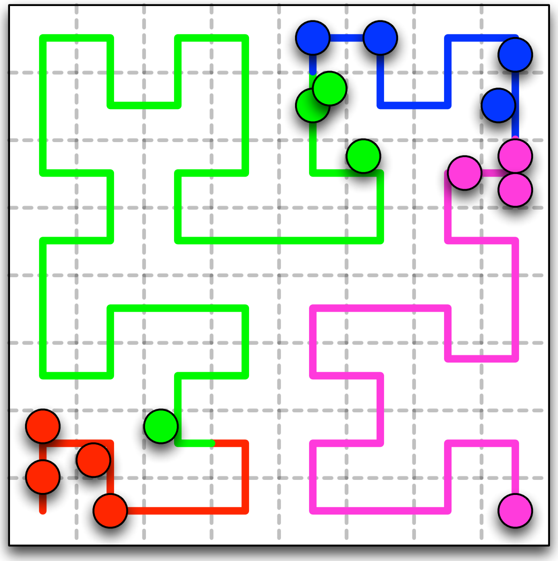
NSF CENTER for CHEMICAL INNOVATION

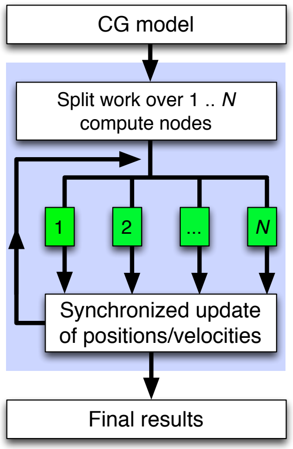
Load balancing

Example: 16 particle simulation run with four CPUs:


CG model
Split work over 1 .. N compute nodes
1
2
...
N
Synchronized update of positions/velocities
Final results






CG model
Split work over 1 .. N compute nodes
1
2
...
N
Synchronized update of positions/velocities
Final results

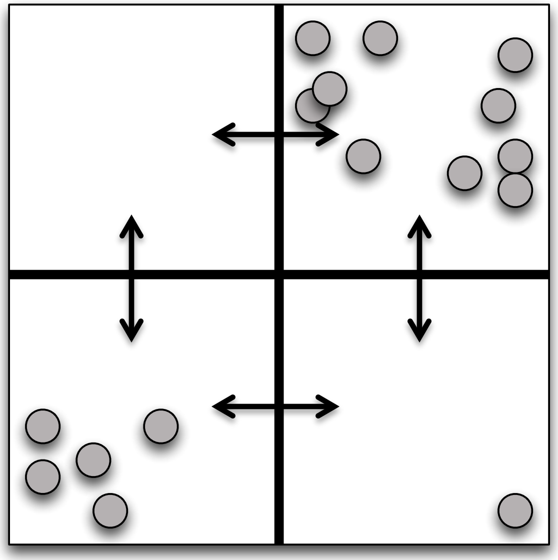
Naïve, uniform spatial decomposition

Hilbert curve spatial decomposition

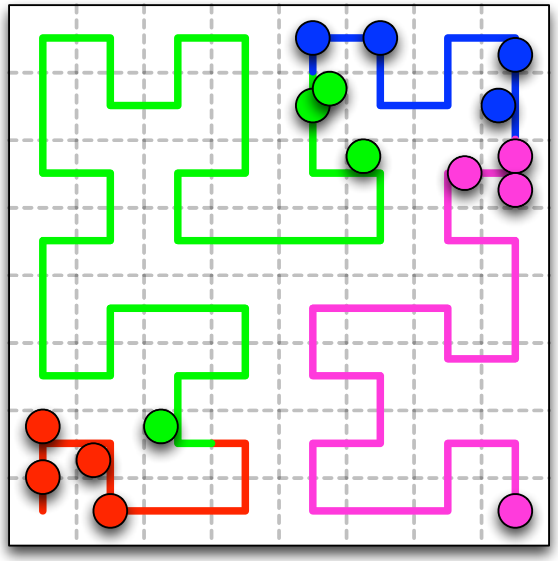
CENTER for MULTISCALE THEORY and SIMULATION
NSF CENTER for CHEMICAL INNOVATION

Load balancing

Who do we need to talk to?



Conventional: easy! We have up, down, left, right … etc

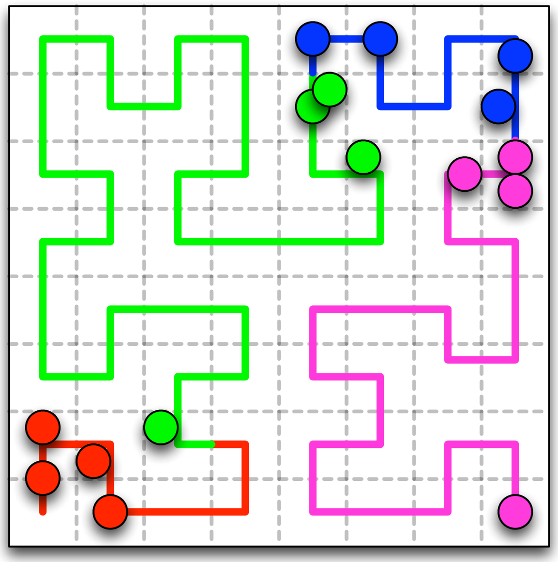


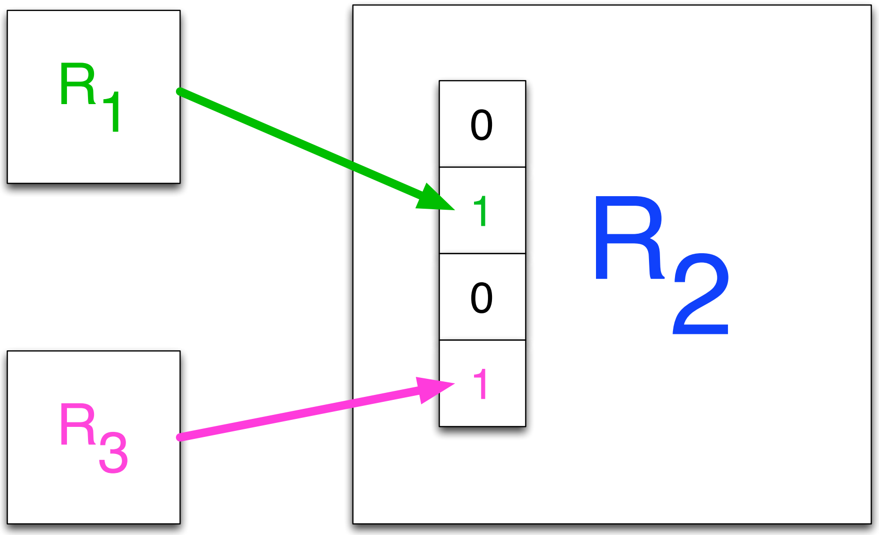
Hilbert curve: Irregular, boundaries between domains not simple

CENTER for MULTISCALE THEORY and SIMULATION
NSF CENTER for CHEMICAL INNOVATION

Load balancing

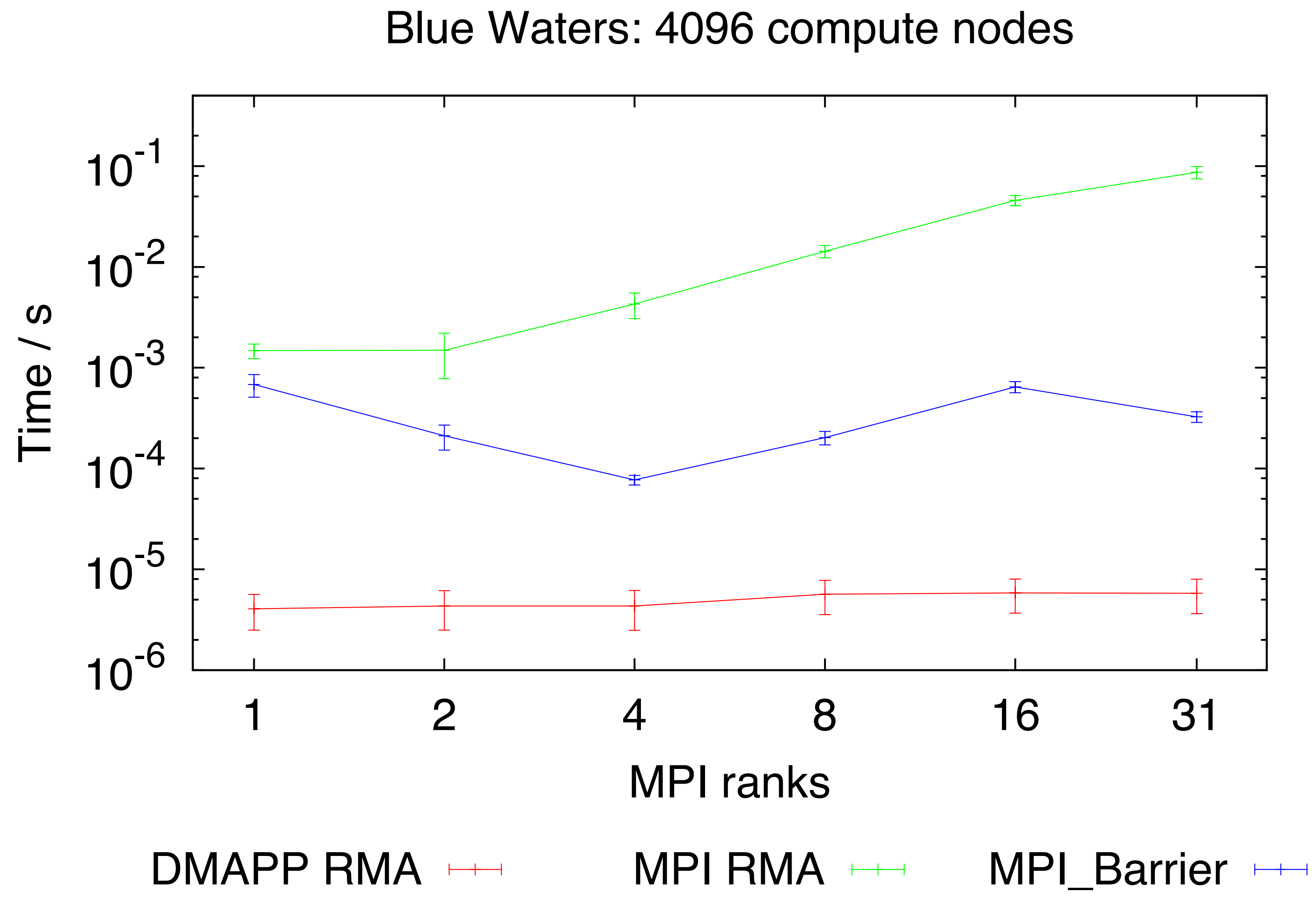
Who do we need to talk to? *MPI_Allreduce* etc slow.





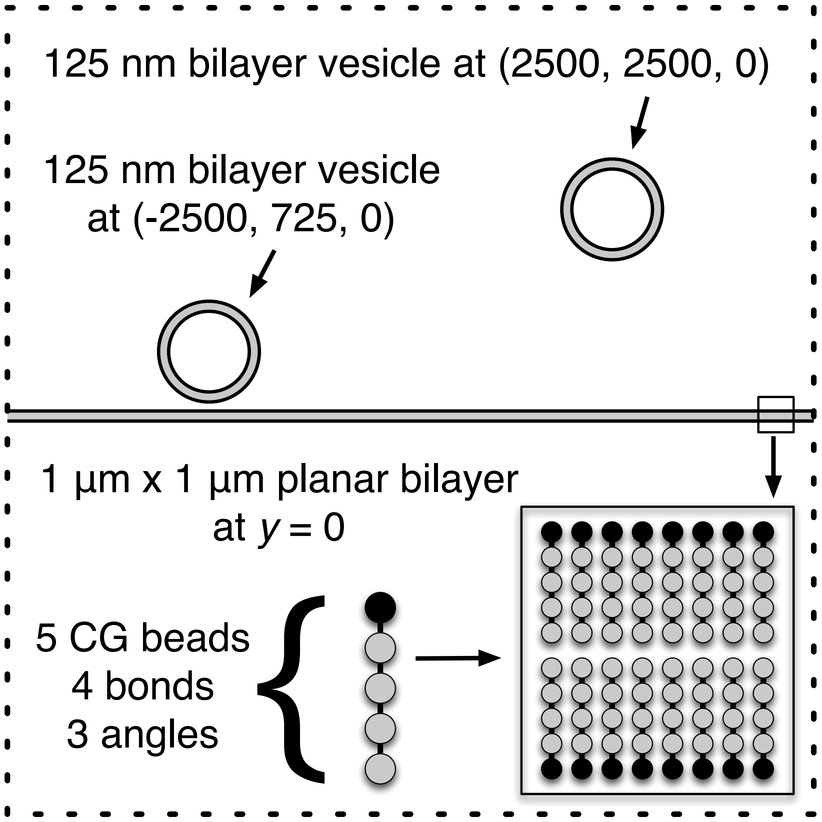
MPI's *Remote Memory Access* (RMA) to the rescue!

Load balancing

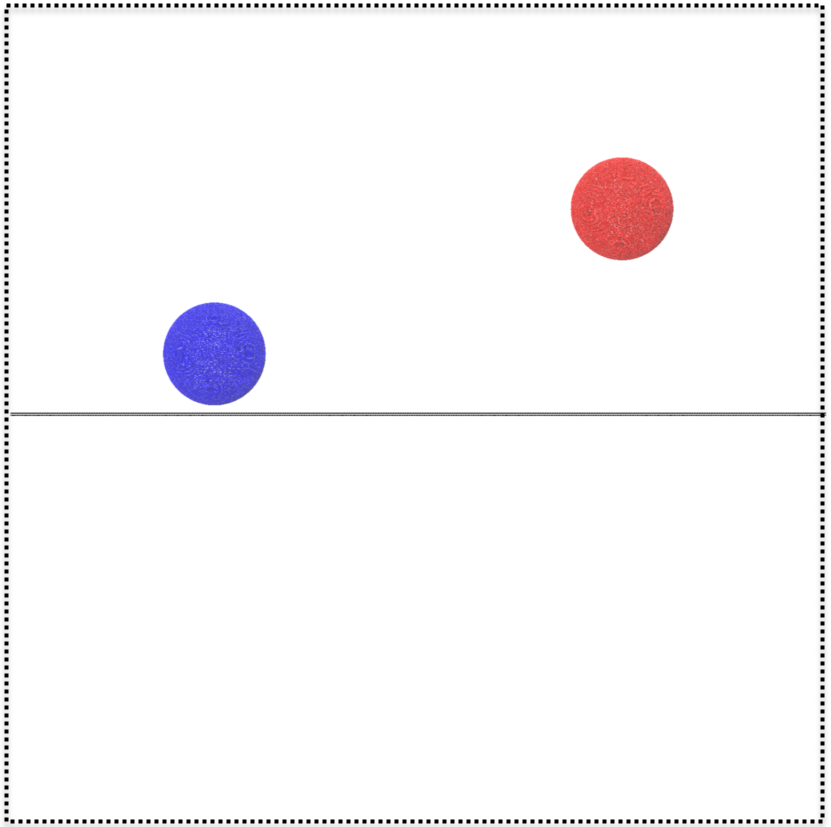


… or maybe not. I used DMAPP!

Example test system



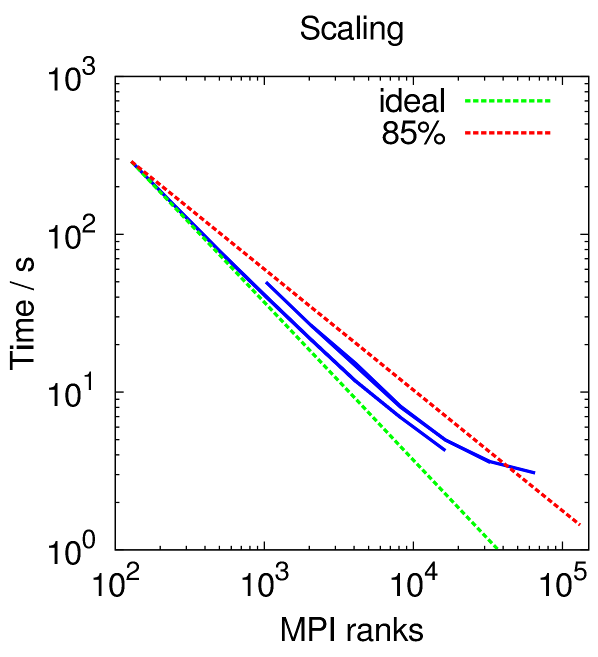
125 nm bilayer vesicle at (2500, 2500, 0)
125 nm bilayer vesicle at (-2500, 725, 0)
1 μm x 1 μm planar bilayer at $y = 0$
5 CG beads
4 bonds
3 angles

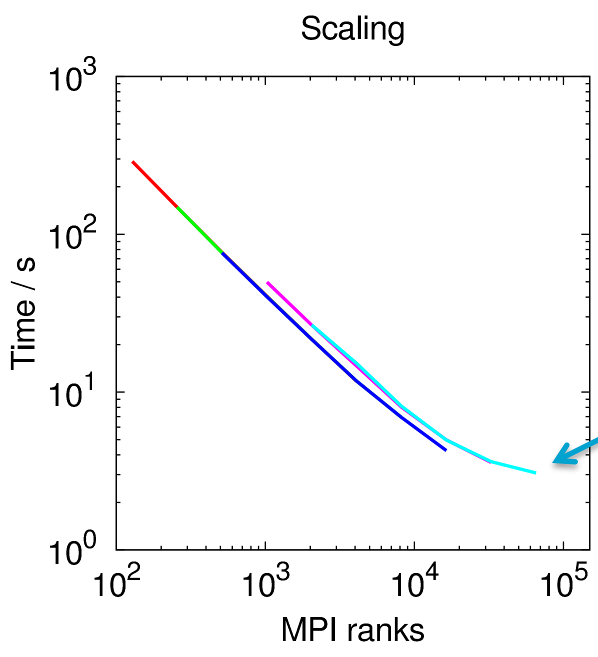


Quite small (~15M particles), ***extremely*** heterogeneous density

THE UNIVERSITY OF CHICAGO

Example test system



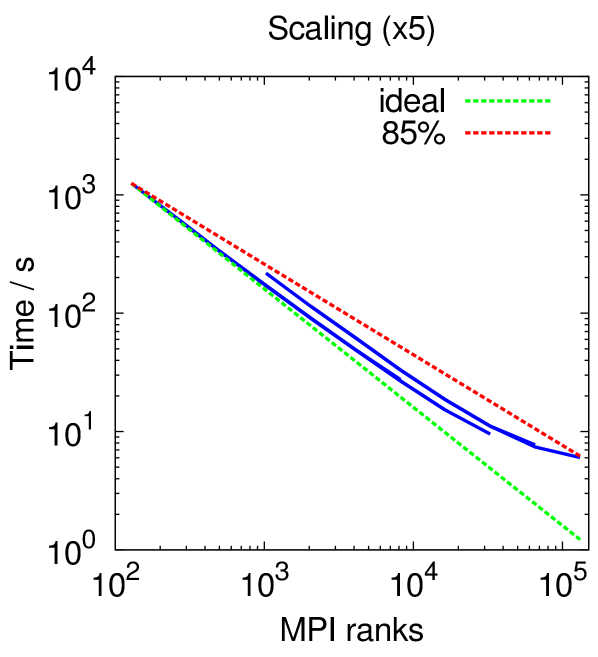


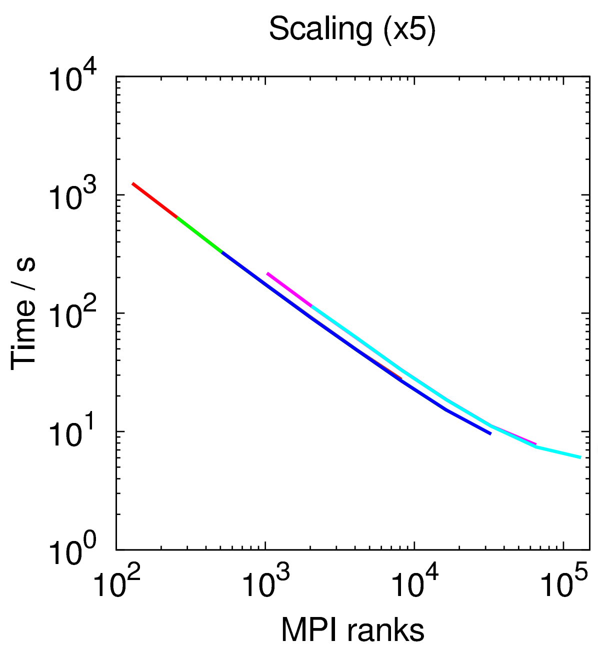
238 particles per rank, ~3 ms per MD timestep

128, 256, … 4096 compute nodes with 1, 2, 4, 8, 16 ranks per node. *Why does it stop scaling?*

CENTER for MULTISCALE THEORY and SIMULATION
NSF CENTER for CHEMICAL INNOVATION

Example test system

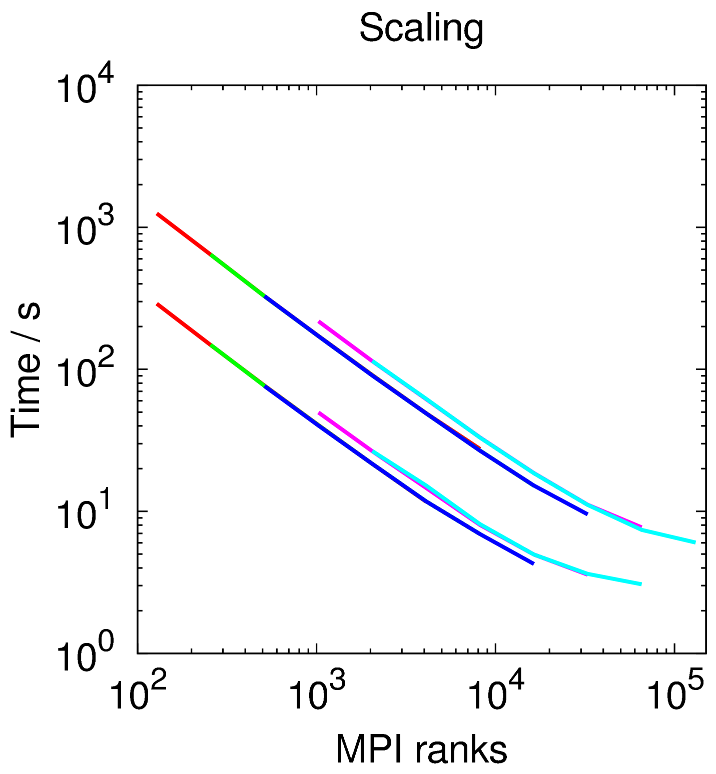




119 particles per rank, ~6 ms per MD timestep

Repeat force calculation x5 at each timestep: different scaling characteristics! *It's not the load balancer!*

Example test system

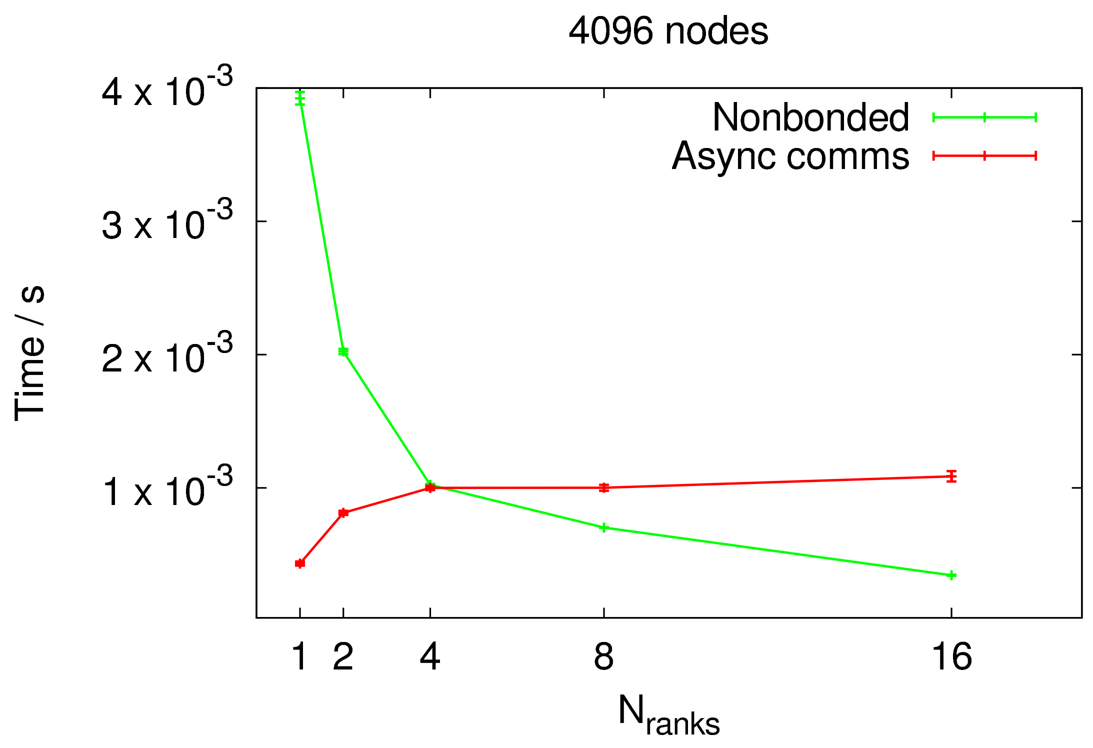


Not simply the same graph shifted up/down; so what's the problem? *It's not due to an MPI collective!*

CENTER for MULTISCALE THEORY and SIMULATION

NSF CENTER for CHEMICAL INNOVATION

Example test system



Problem is MPI's asynchronous point-to-point calls (irecv, isend) – they can't keep up! DMAPP again?

THE UNIVERSITY OF CHICAGO

Summary

- Designed & implemented a custom MD code for large-scale, very dynamic CG simulations:
    - Very memory efficient
    - Load balances well
    - “Simple” & relatively friendly; < 5K lines of C++ with vanilla MPI (except for a few lines of DMAPP RMA code)

- Performance is very promising, despite unoptimised state

CENTER for MULTISCALE THEORY and SIMULATION
NSF CENTER for CHEMICAL INNOVATION

THE UNIVERSITY OF CHICAGO

Thanks to …

- NSF
- Prof. Greg Voth & the group
- Blue Waters staff, particularly:
  - Robert Brunner (long-suffering point-of-contact)
  - Kalyana Chadalavada

- jgrime@uchicago.edu - **comments and advice are very welcome!**

CENTER for MULTISCALE THEORY and SIMULATION
NSF CENTER for CHEMICAL INNOVATION