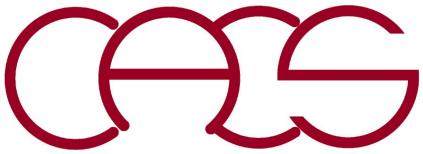
Advanced Topics in Parallel Molecular Dynamics

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Quantitative & Computational Biology
University of Southern California

Email: anakano@usc.edu



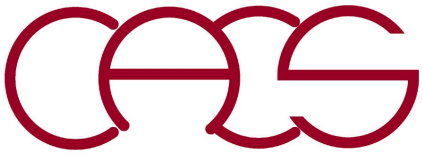
cf. https://aiichironakano.github.io/cs653.html

Load Balancing

Aiichiro Nakano

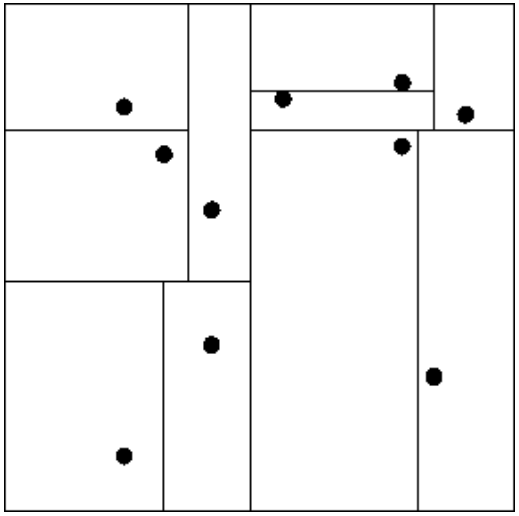
Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Quantitative & Computational Biology
University of Southern California

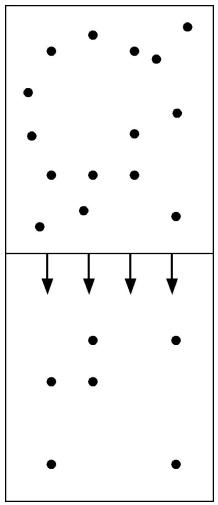
Email: anakano@usc.edu

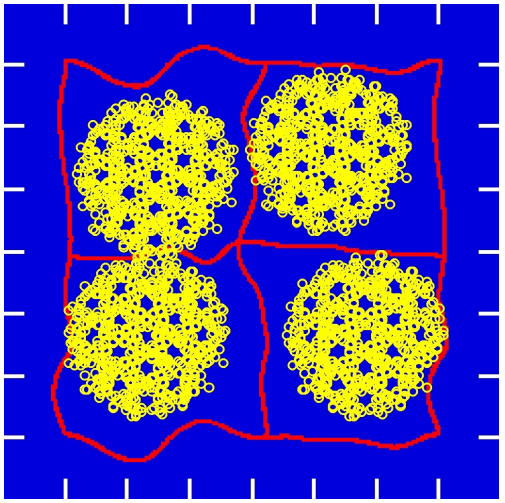


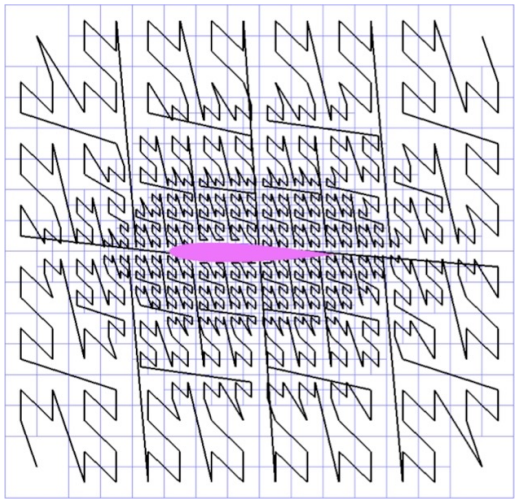
Load Balancing

- **Goal:** Keep all processors equally busy while minimizing inter-processor communication for irregular parallel computations
- **Issues:**
  - Spatial data *vs.* generic graph
  - Static *vs.* adaptive
  - Incremental *vs.* non-incremental
- **Load-balancing schemes:**
  - Recursive bisection
  - Spectral method
  - Spacefilling curve
  - Curved space
  - Load diffusion





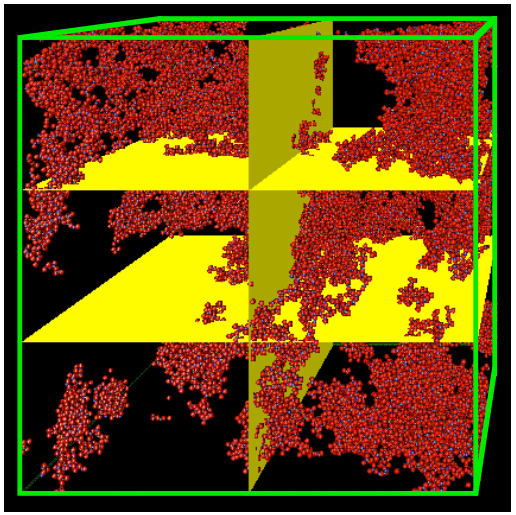




Data Locality in Parallelization

Challenge: Load balancing for irregular data structures

Irregular data-structures/ processor-speed

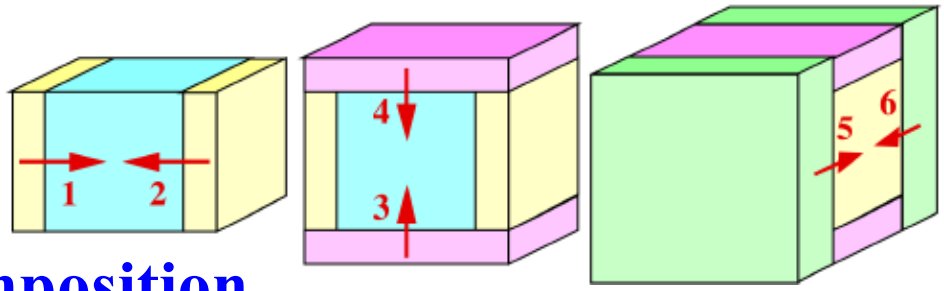


Map

Parallel computer

Optimization problem:

- Minimize the load-imbalance cost
- Minimize the communication cost
- Topology-preserving spatial decomposition
  → structured 6-step message passing minimizes latency



$$E = t_{\text{comp}}\left(\max_p \left|\{i \mid \mathbf{r}_i \in p\}\right|\right) + t_{\text{comm}}\left(\max_p \left|\{i \mid \|\mathbf{r}_i - \partial p\| < r_c\}\right|\right)$$
$$+ t_{\text{latency}}\left(\max_p \left[N_{\text{message}}(p)\right]\right)$$

Computational-Space Decomposition

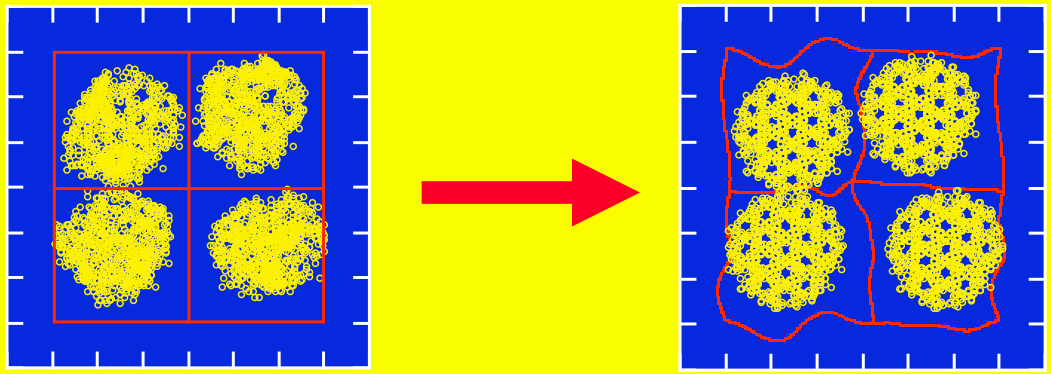
Topology-preserving "computational-space" decomposition in curved space (*cf.* general relativity)

Curvilinear coordinate transformation

$$\xi = x + u(x)$$

Particle-processor mapping: regular 3D mesh topology

$$\begin{cases} p(\xi_i) = p_x(\xi_{ix}) P_y P_z + p_y(\xi_{iy}) P_z + p_z(\xi_{iz}) \\ p_\alpha(\xi_{i\alpha}) = \lfloor \xi_{i\alpha} P_\alpha / L_\alpha \rfloor \quad (\alpha = x, y, z) \end{cases}$$



Regular mesh topology in computational space, ξ

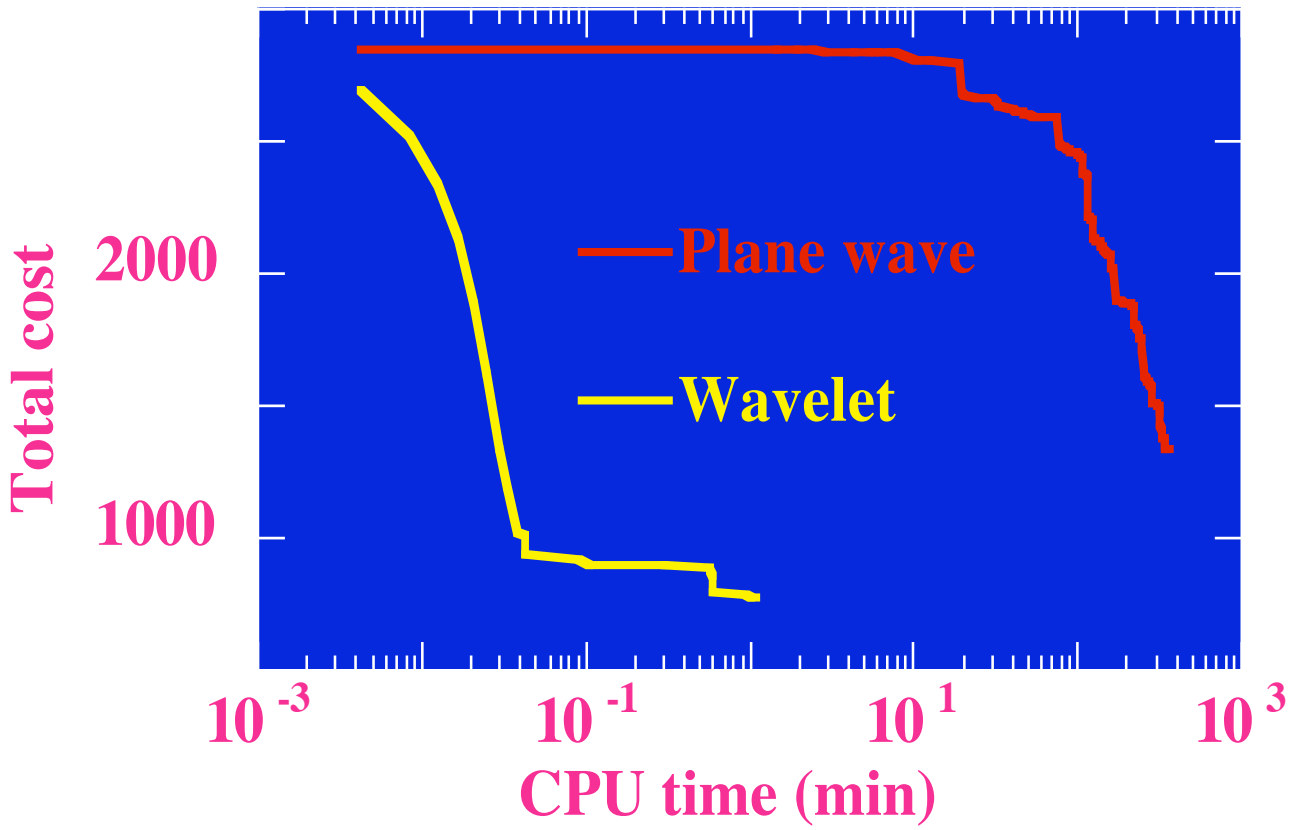
Curved partition in physical space, x

A. Nakano & T. J. Campbell, *Parallel Comput.* **23**, 1461 ('97)

Wavelet-based Adaptive Load Balancing

- **Simulated annealing to minimize the load-imbalance & communication costs, $E[\xi(x)]$**
- **Wavelet representation speeds up the optimization**

$$\xi(x) = x + \sum_{l,m} d_{lm}\psi_{lm}(x)$$

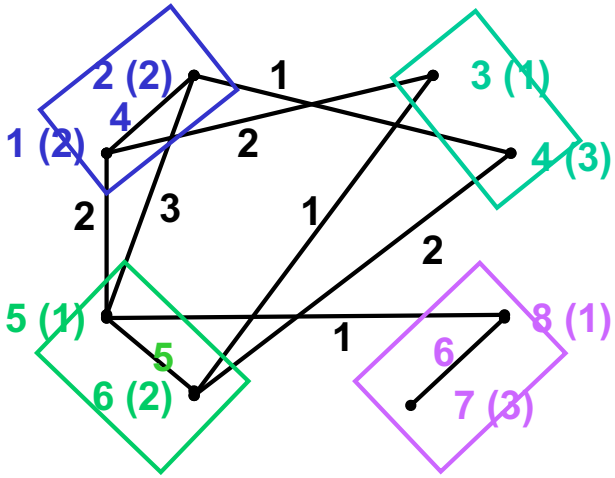


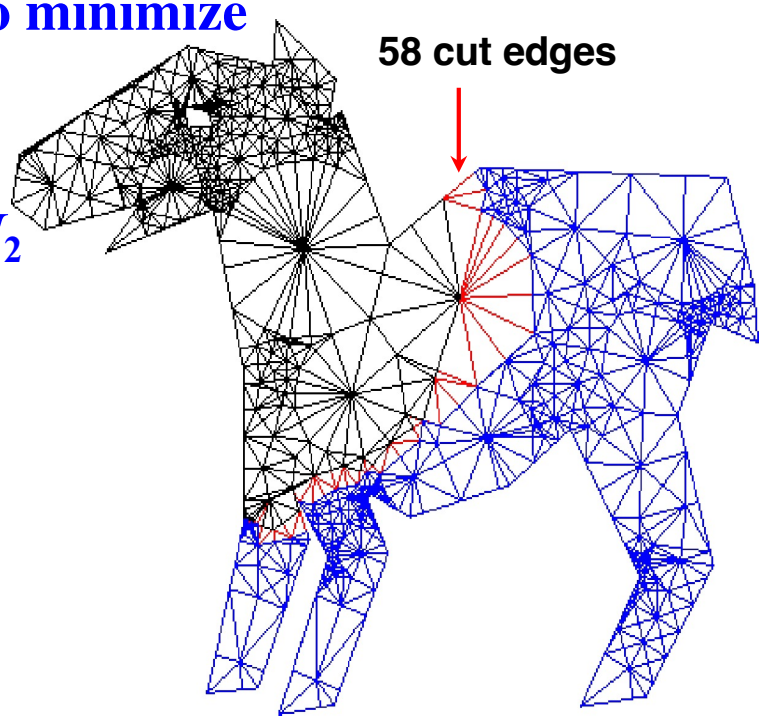
A. Nakano, *Concurrency: Practice and Experience* **11**, 343 ('99)

Load Balancing as Graph Partitioning

- **Need: Decompose tasks without spatial indices**
- **Graph partitioning: Given a graph $G = (N, E, W_N, W_E)$**
  - **N: node set = $\{j \mid$ tasks$\}$**
  - **W_N: node weights = $\{w_N(j)$: task costs$\}$**
  - **E: edge set = $\{(j,k) \mid$ messages from j to $k\}$**
  - **W_E: edge weights = $\{w_E(j,k)$: message sizes$\}$**

  choose a partition $N = N_1 \cup N_2 \cup \ldots \cup N_P$ to minimize
  - **$\max_p\{\sum_{j \in Np} w_N(j)\}$**
  - **$\max_{(p,q)}\{\sum_{j \in Np, k \in Nq} w_E(j,k)\}$**
- **Graph bisection: Special case of $N = N_1 \cup N_2$**
- **Choosing optimal partitioning is known to be NP-complete → need heuristics**



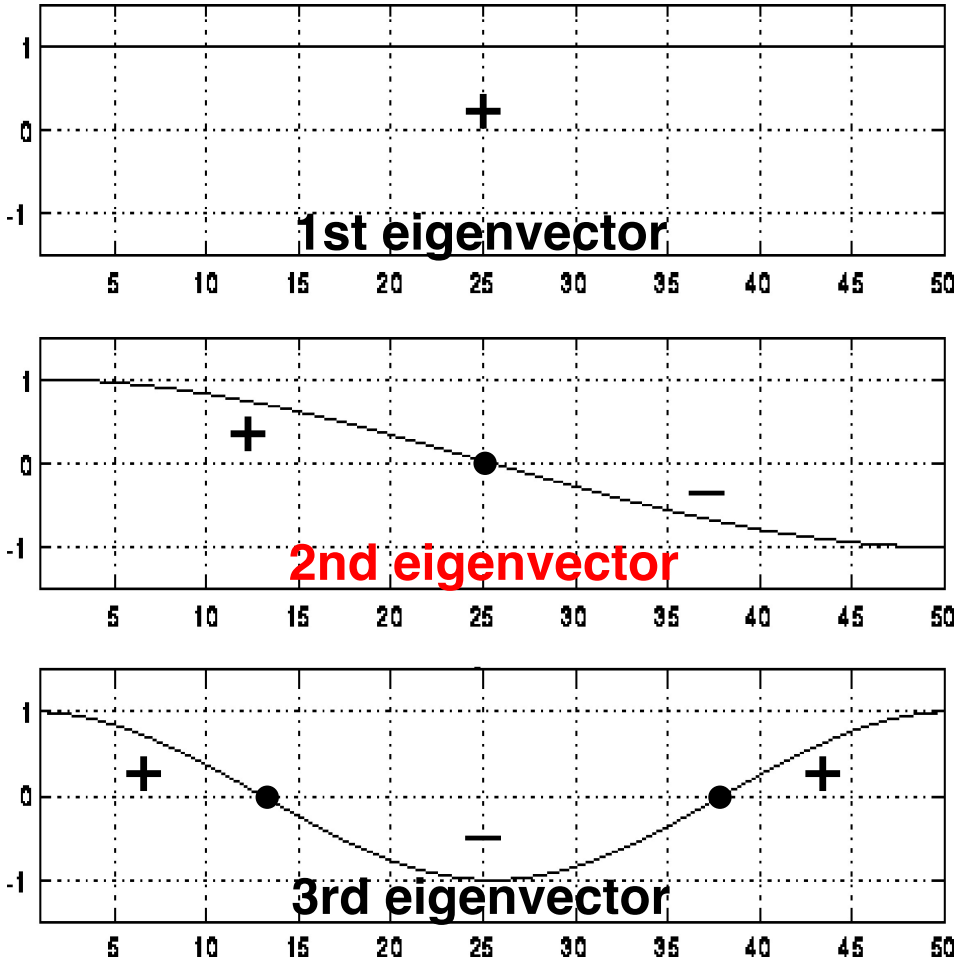


Prof. James Demmel (UC Berkeley)

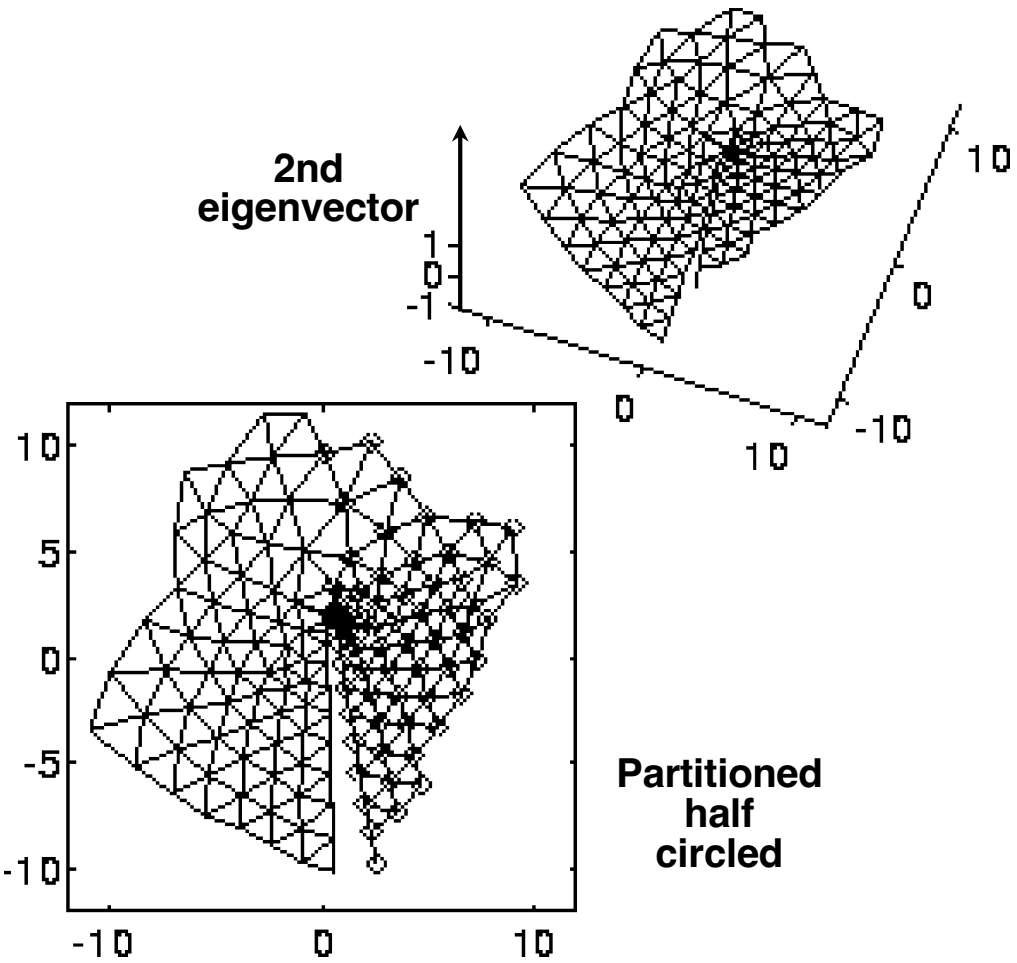
Spectral Bisection: Motivation

1. Graph as point masses connected *via* harmonic springs
2. The node of the eigenvector of the Hessian matrix, $\partial^2/\partial x^2$, corresponding to the 2nd smallest eigenvalue separates the graph into 2

1D example



2D example



Spectral Bisection

Laplacian matrix:

L(G) of a graph $G(N,E)$ is an $|N|$ by $|N|$ symmetric matrix:

- L(G)(i,i) = degree of node i (number of incident edges)
- L(G)(i,j) = –1 if $i \neq j$ and there is an edge (i,j)
- L(G)(i,j) = 0 otherwise

Theorems:

1. The eigenvalues of L(G) are nonnegative:

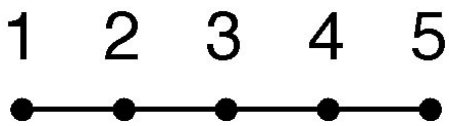
   $\lambda_1 = 0 \leq \lambda_2 \leq \cdots \leq \lambda_N$

2. $\lambda_2(L(G)) \neq 0$ if and only if G is connected

Spectral bisection algorithm:

1. Compute eigenvector v_2 corresponding to $\lambda_2(L(G))$
2. For each node i of G
   a. if $v_2(i) < 0$, put node i in partition N_-
   b. else put node i in partition N_+

Example



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix} \end{array}$$

$O(N)$ λ_2 Computation

Lanczos algorithm:

- **Given an $N \times N$ symmetric matrix A (*e.g.*, L(G)), compute a $K \times K$ "approximation" T by performing K matrix-vector products, where $K \ll N$**
- **Approximate A's eigenvalues & eigenvectors using T's**

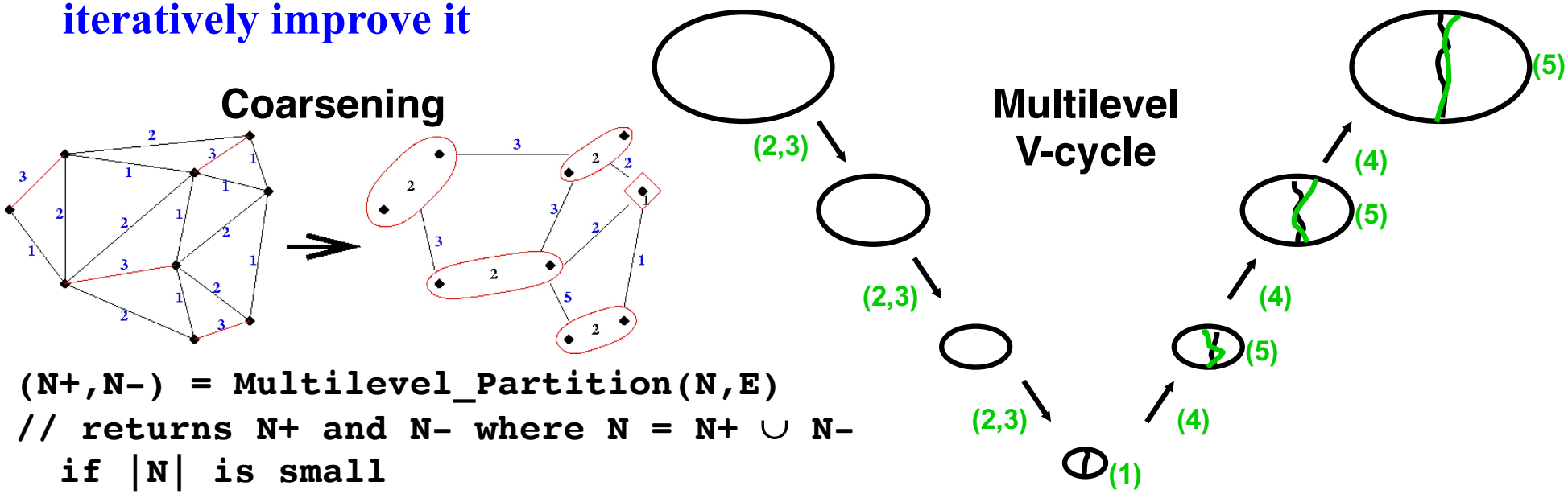
```
Choose an arbitrary starting vector r
b(0) = ||r||
j=0
repeat
  j=j+1
  q(j) = r/b(j-1)
  r = A*q(j)
  r = r - b(j-1)*v(j-1)
  a(j) = v(j)^T * r
  r = r - a(j)*v(j)
  b(j) = ||r||
until convergence
```

$$\mathbf{T} = \begin{bmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{K-2} & a_{K-1} & b_{K-1} \\ & & & b_{K-1} & a_K \end{bmatrix}$$

Multilevel Partitioning

Recursively apply:

1. Replace $G(N,E)$ by a coarse approximation $G_c(N_c,E_c)$, & partition G_c
2. Use partition of G_c to obtain a rough partitioning of G, then uncoarsen & iteratively improve it



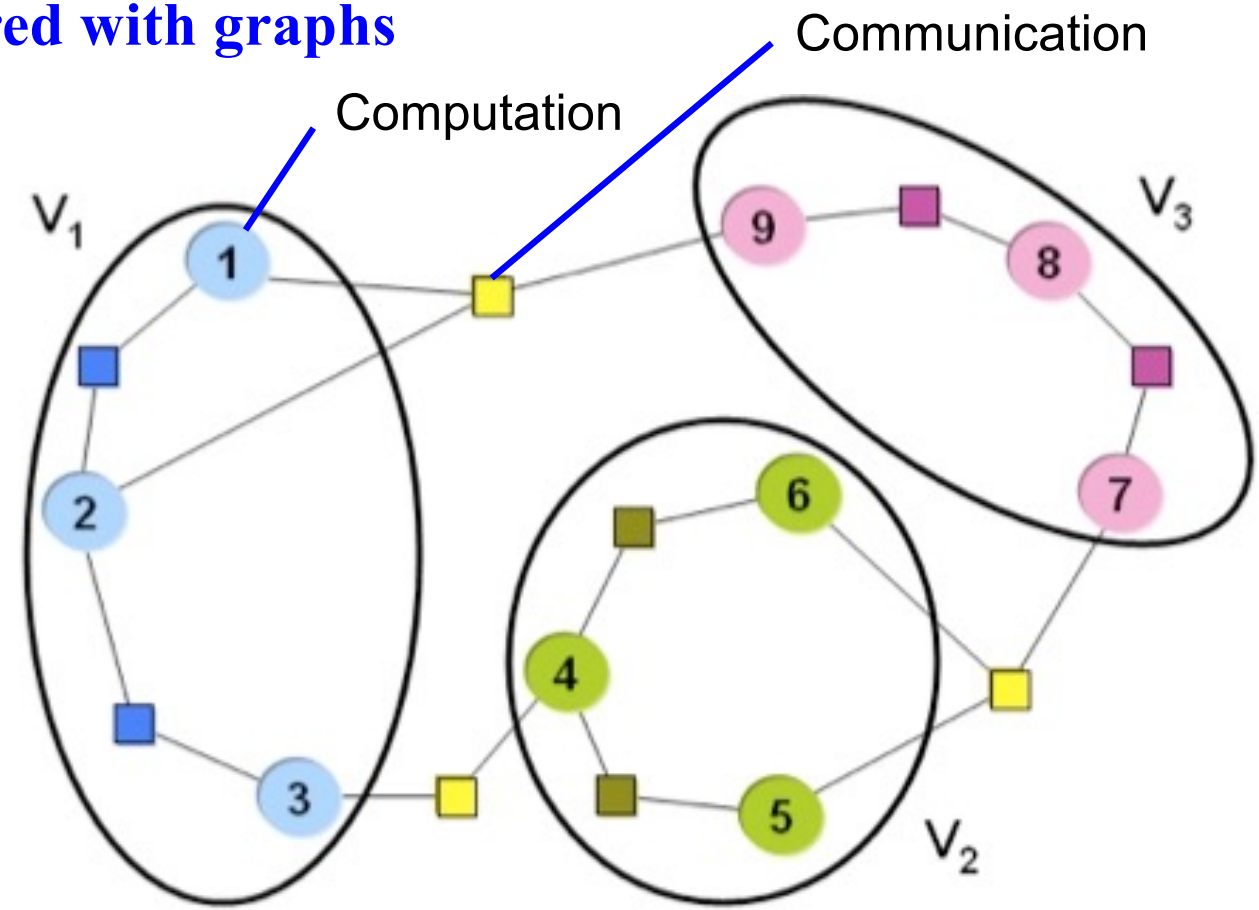

```
(N+,N-) = Multilevel_Partition(N,E)
// returns N+ and N- where N = N+ ∪ N-
  if |N| is small
1   Partition G = (N,E) directly to get N = N+ ∪ N-
    Return (N+,N-)
  else
2   Coarsen G to get an approximation Gc = (Nc,Ec)
3   (Nc+,Nc-) = Multilevel_Partition(Nc,Ec)
4   Expand (Nc+,Nc-) to a partition (N+,N-) of N
5   Improve the partition (N+,N-)
    Return (N+,N-)
  endif
```

cf. Multigrid method

cf. Shang-Hua Teng, https://dl.acm.org/doi/10.1145/3627708

Hypergraph-based Load Balancing

1. **Hypergraph = ({node}, {hyperedge = a group of nodes})**
2. **More expressive power for computation-communication relation compared with graphs**



U. V. Catalyurek *et al*., "Hypergraph-based dynamic load balancing for adaptive scientific computations," in *Proc. IPDPS* (IEEE, '07)

M. Kunaseth *et al*., "A scalable parallel algorithm for dynamic range-limited *n*-tuple computation in many-body molecular dynamics simulation," in *Proc. SC* (ACM/IEEE, '13)

Hybrid Decomposition

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Quantitative & Computational Biology
University of Southern California

Email: anakano@usc.edu

Who does what?

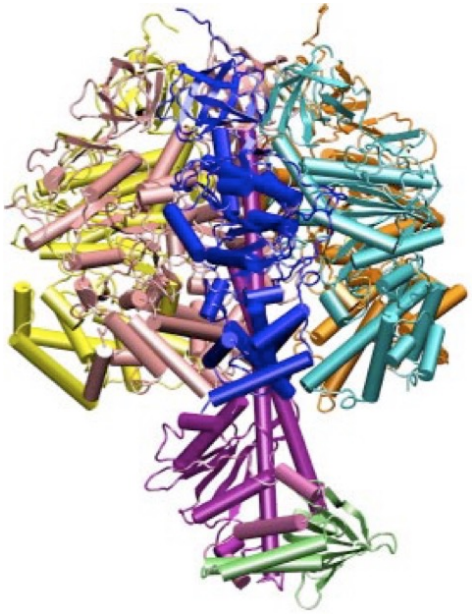
Fine-Grained Parallel MD

Pathways to a Protein Folding Intermediate Observed in a 1-Microsecond Simulation in Aqueous Solution

Yong Duan and Peter A. Kollman*

An implementation of classical molecular dynamics on parallel computers of increased efficiency has enabled a simulation of protein folding with explicit representation of water for 1 microsecond, about two orders of magnitude longer than the longest simulation of a protein in water reported to date. Starting with an unfolded state of villin headpiece subdomain, hydrophobic collapse and helix formation occur in an initial phase, followed by conformational readjustments. A marginally stable state, which has a lifetime of about 150 nanoseconds, a favorable solvation free energy, and shows significant resemblance to the native structure, is observed; two pathways to this state have been found.

Science **282**, 740 ('98)

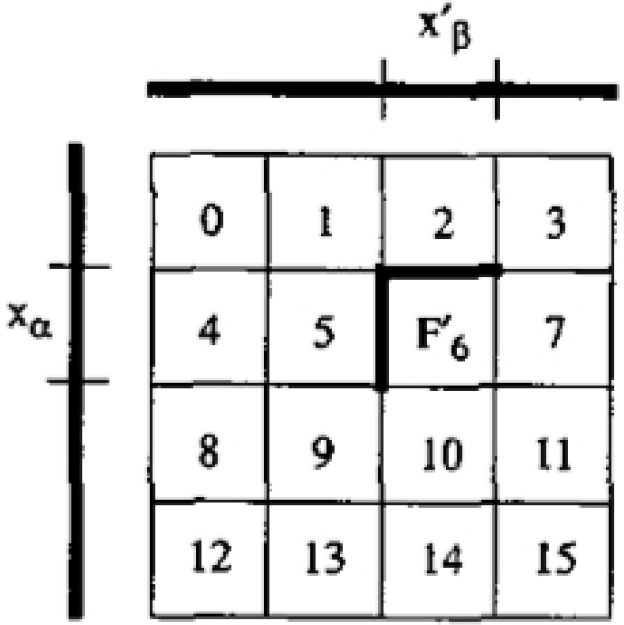


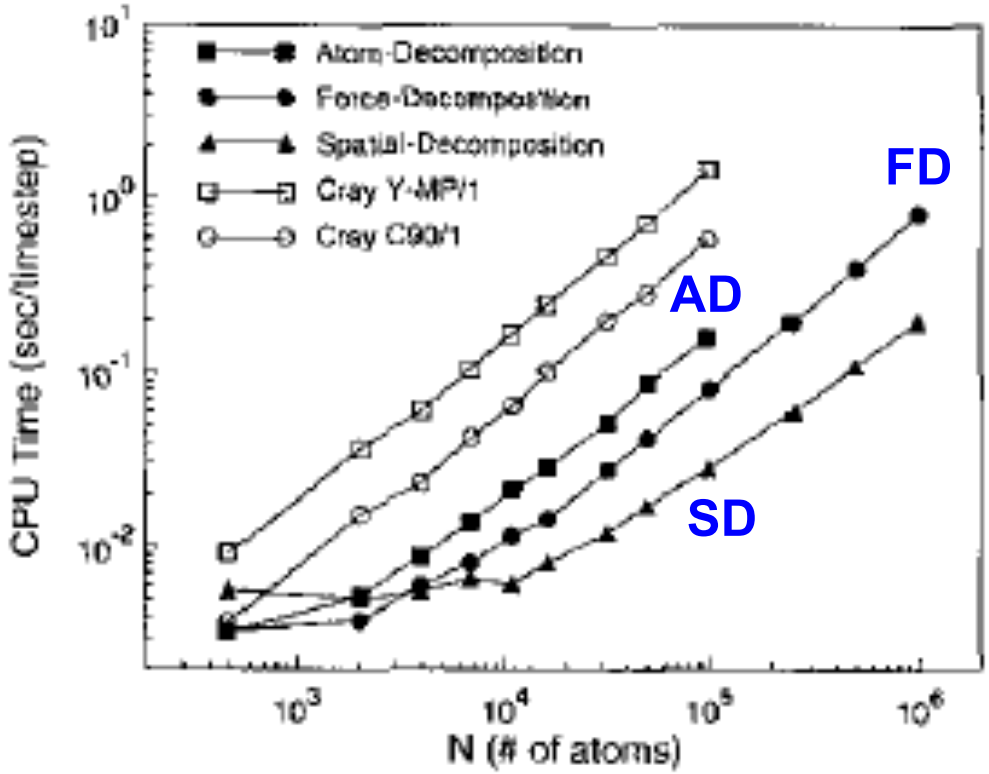
Processors		Time/step		Speedup		GFLOPS	
Total	Per Node	MPI	Elan	MPI	Elan	MPI	Elan
1	1	28.08 s	28.08 s	1	1	0.480	0.480
128	4	248.3 ms	234.6 ms	113	119	54	57
256	4	135.2 ms	121.9 ms	207	230	99	110
512	4	65.8 ms	63.8 ms	426	440	204	211
510	3	65.7 ms	63.0 ms	427	445	205	213
1024	4	41.9 ms	36.1 ms	670	778	322	373
1023	3	35.1 ms	33.9 ms	799	829	383	397
1536	4	35.4 ms	32.9 ms	792	854	380	410
1536	3	26.7 ms	24.7 ms	1050	1137	504	545
2048	4	31.8 ms	25.9 ms	883	1083	423	520
1800	3	25.8 ms	22.3 ms	1087	1261	521	605
2250	3	19.7 ms	18.4 ms	1425	1527	684	733
2400	4	32.4 ms	27.2 ms	866	1032	416	495
2800	4	32.3 ms	32.1 ms	869	873	417	419
3000	4	32.5 ms	28.8 ms	862	973	414	467

Table 1: NAMD performance on 327K atom ATPase benchmark system with and multiple timestepping with PME every four steps for Charm++ based on MPI and Elan.

J.C. Phillips, G. Zheng, S. Kumar, & L.V. Kale,
in *Proc. of IEEE/ACM SC2002*

Force Decomposition for Parallel MD





Runtime on 1,024-processor Intel Paragon

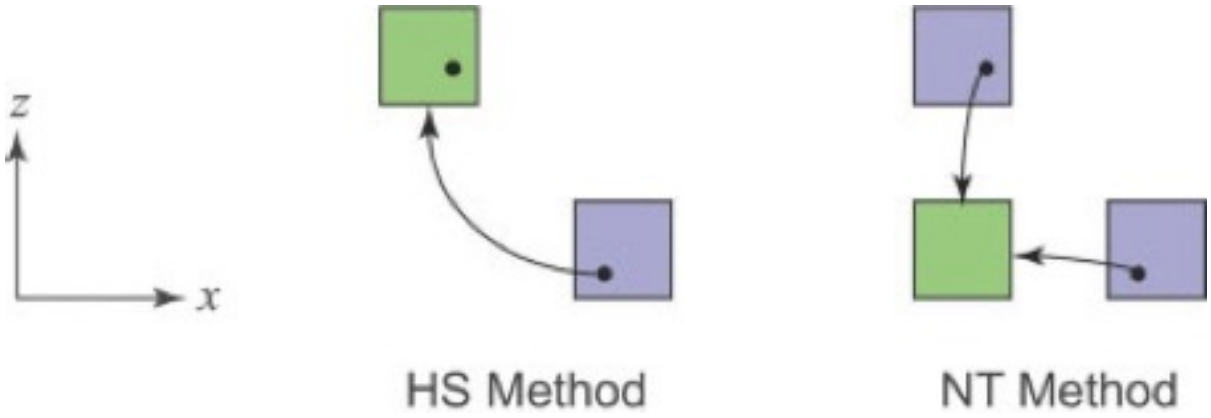
FIG. 5. The division of the permuted force matrix F' among 16 processors in the force-decomposition algorithm. Processor P_6 is assigned a sub-block F'_6 of size $N/\sqrt{P}$ by $N/\sqrt{P}$. To compute its matrix elements it must know the corresponding $N/\sqrt{P}$-length pieces x_α and x'_β of the position vector x and permuted position vector x'.

S. Plimpton, *J. Comput. Phys.* **117**, 1 ('95)

https://www.lammps.org/cite.html

Neutral Territory Decomposition

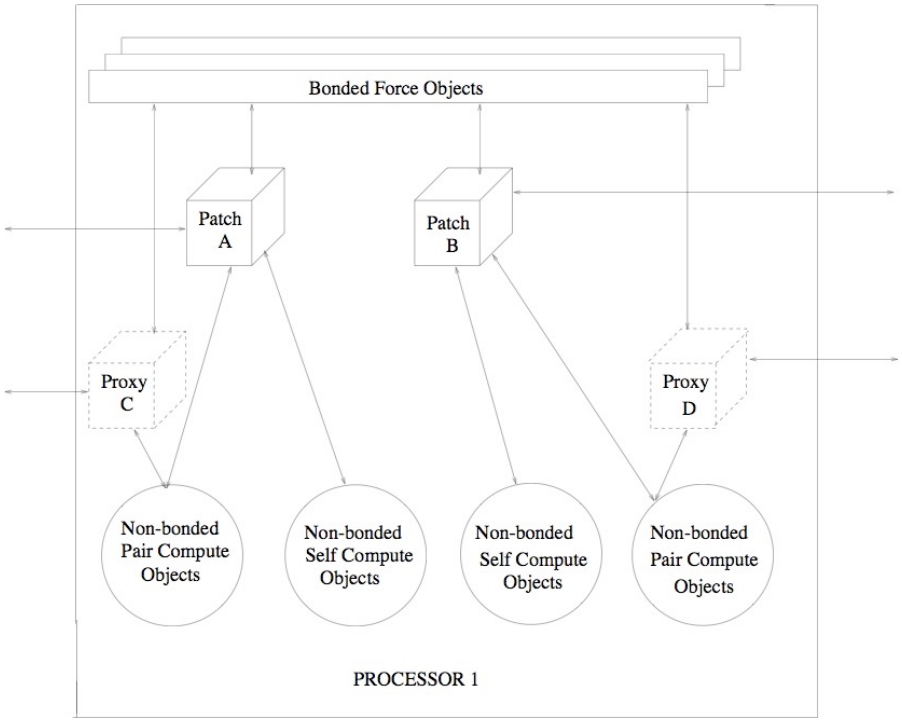
D. E. Shaw,
"A fast, scalable method for the parallel evaluation of distance-limited pairwise particle interactions,"
J. Comput. Chem. **26**, 1318 ('05)

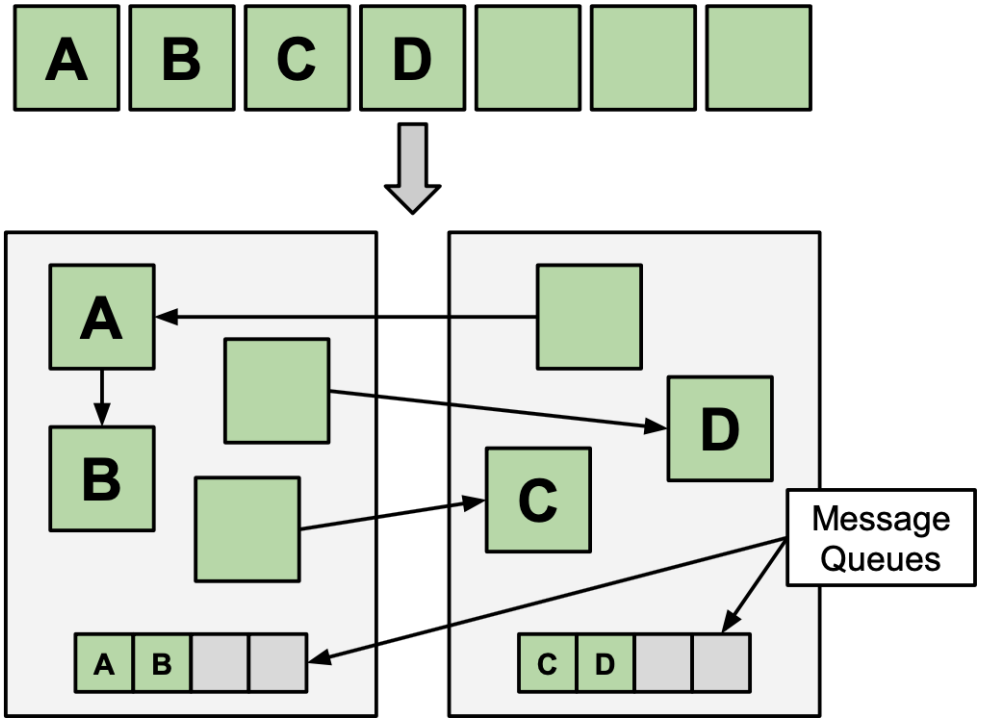


***cf.* Lecture note on "Shaw's NT algorithm"**

Hybrid Spatial+Force Decomposition

- **Spatial decomposition of patches (localized spatial regions & atoms within)**
- **Inter-patch force computation objects assigned to any processor**
- **Message-driven object execution: computation-communication overlap**





Kale *et al.*, *J. Comput. Phys.* **151**, 283 ('99); Phillips *et al.*, *SC02* (IEEE/ACM); Acun *et al.*, *SC14* (IEEE/ACM), Phillips *et al.*, *J. Chem. Phys.* 153, 044130 ('20)