

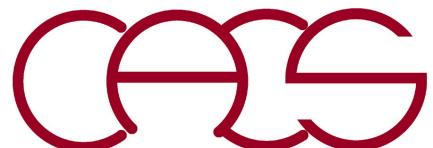
Divide-&-Conquer Parallelism

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

Only one thing: Most scalable algorithmic paradigm into future

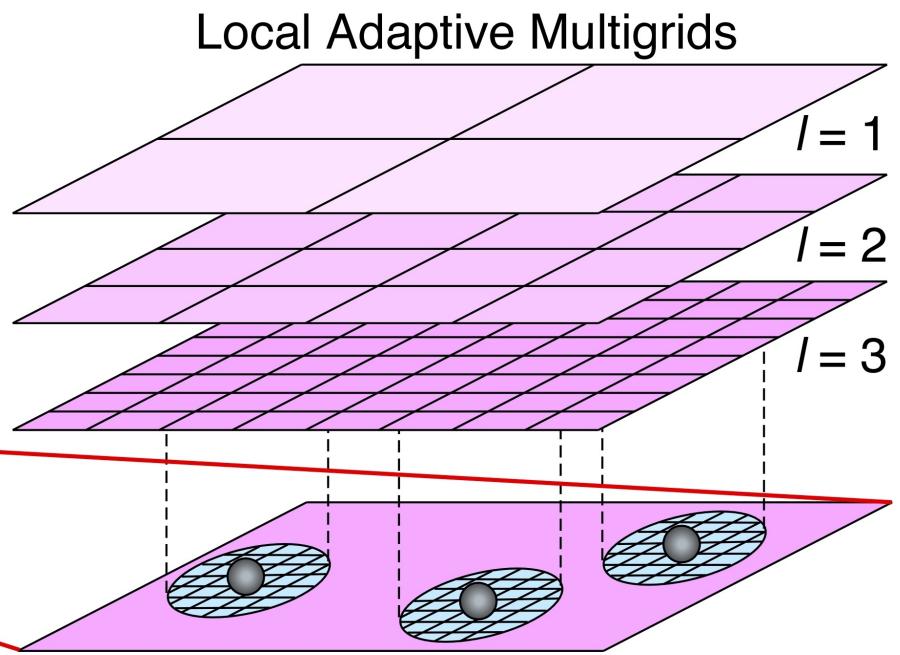
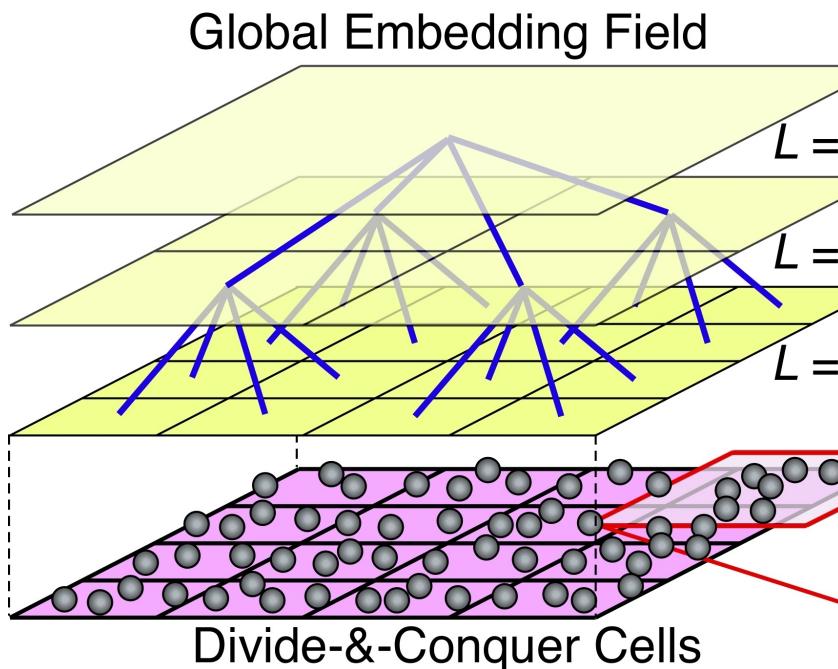


cf. <https://aiichironakano.github.io/cs653.html>



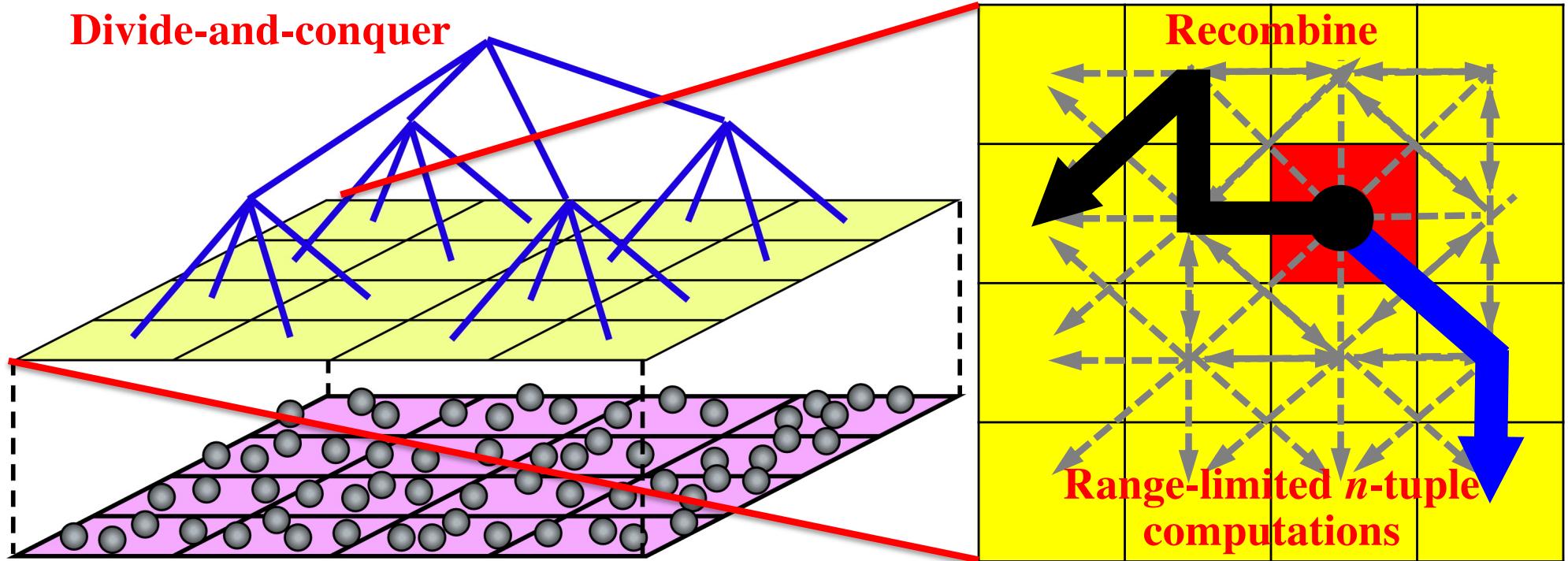
Divide-&-Conquer Algorithms

DC for (1) $O(N)$ algorithms & (2) scalability $> P = 10^5$



- **N -body problem:** $O(N^2) \rightarrow O(N)$
 - > **Space-time multiresolution molecular dynamics (MRMD):** Fast multipole method & symplectic multiple time stepping
- **Variable N -charge problem:** $O(N^3) \rightarrow O(N)$
 - > **Fast reactive force-field (F-ReaxFF) MD:** Multilevel preconditioning
- **Quantum N -body problem:** $O(C^N) \rightarrow O(N)$
 - > **DC density functional theory (DC-DFT):** Adaptive multigrids

Divide-Conquer-Recombine (DCR)



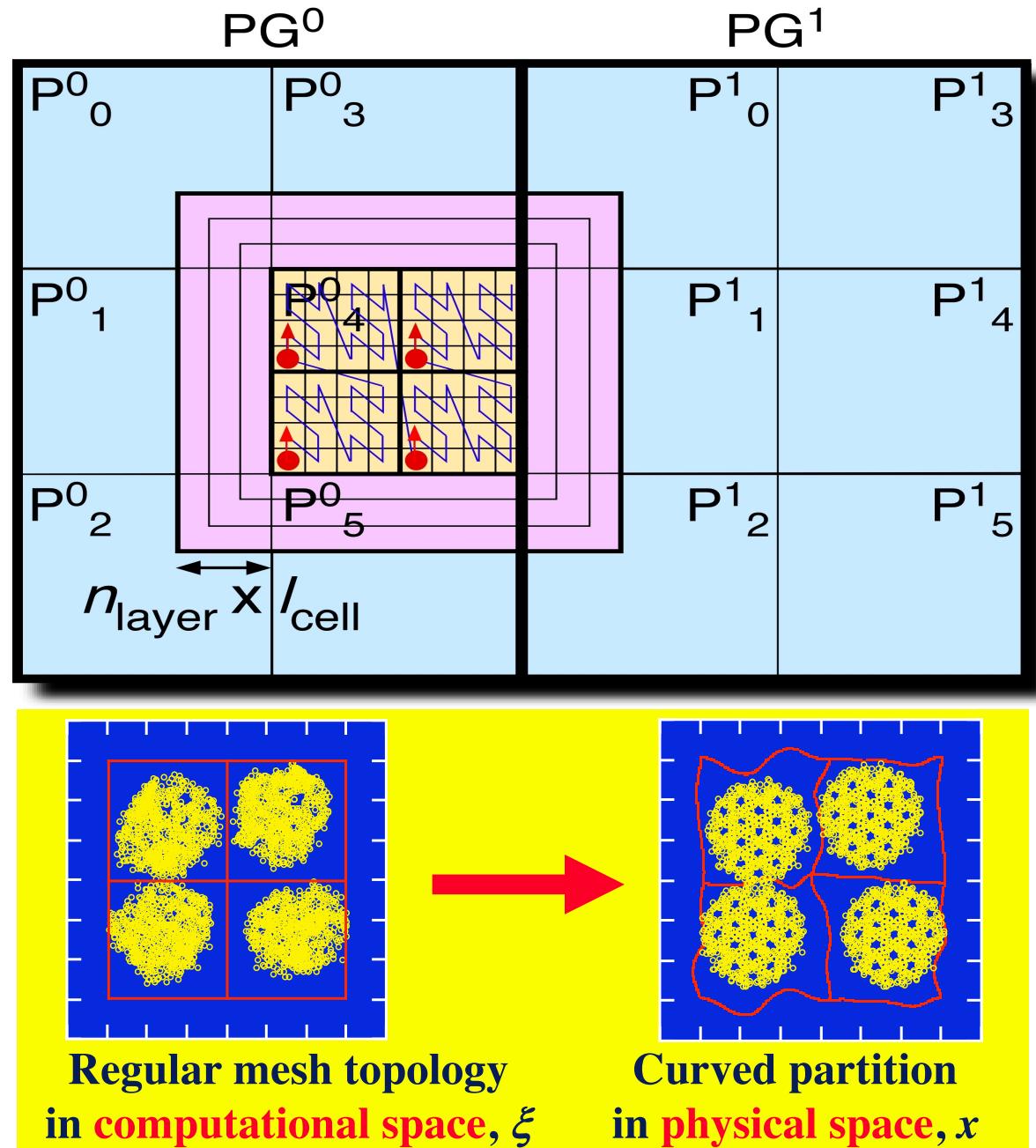
M. Kunaseth et al., ACM/IEEE SC13

- Lean divide-&-conquer density functional theory (LDC-DFT) algorithm minimizes the prefactor of $O(N)$ computational cost in quantum molecular dynamics
Shimojo et al., *J. Chem. Phys.* **140**, 18A529 ('14); Nomura et al., *IEEE/ACM SC14*; Romero et al., *Comput. Sci. & Eng.* **21**, 64 ('19); Tiwari et al., *IEEE HPCAsia20*
- Extended-Lagrangian reactive molecular dynamics (XRMD) algorithm eliminates the speed-limiting charge iteration
Nomura et al., *Comput. Phys. Commun.* **192**, 91 ('15); Liu et al., *IEEE/ACM ScalA18*

Mapping DCR to Hierarchical Hardware

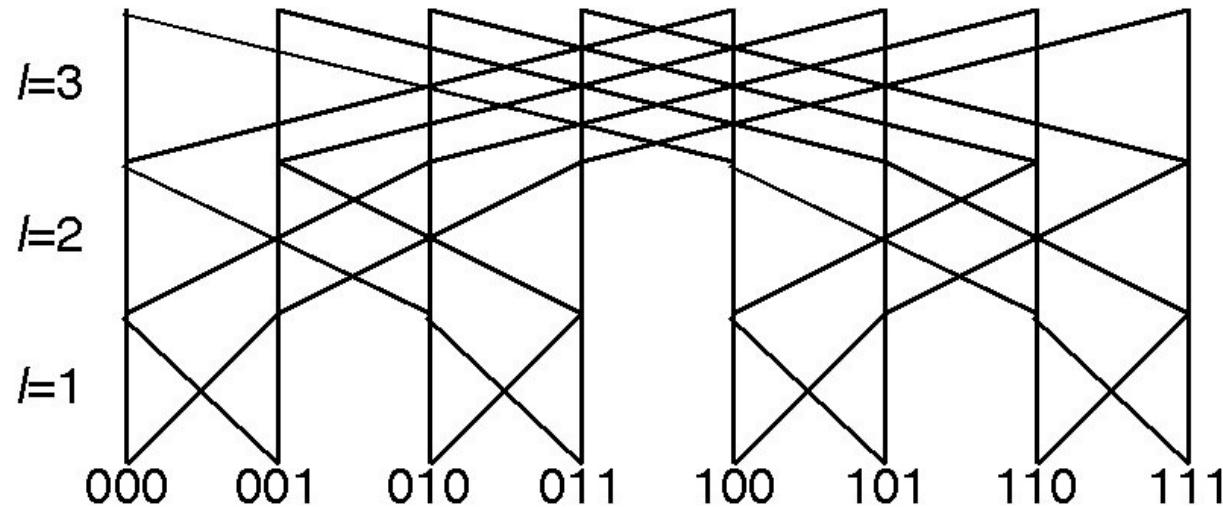
HCD: tunable hierarchical cellular decomposition

- Topology-preserving computational-space decomposition
- Computational cell \subset thread \subset processor \subset processor group
- AI-based computation/ data layout tuning
- Wavelet-based adaptive load balancing
- Spacefilling-curve data compression for I/O
- Hybrid MPI + threads + SIMD (data parallel) programming



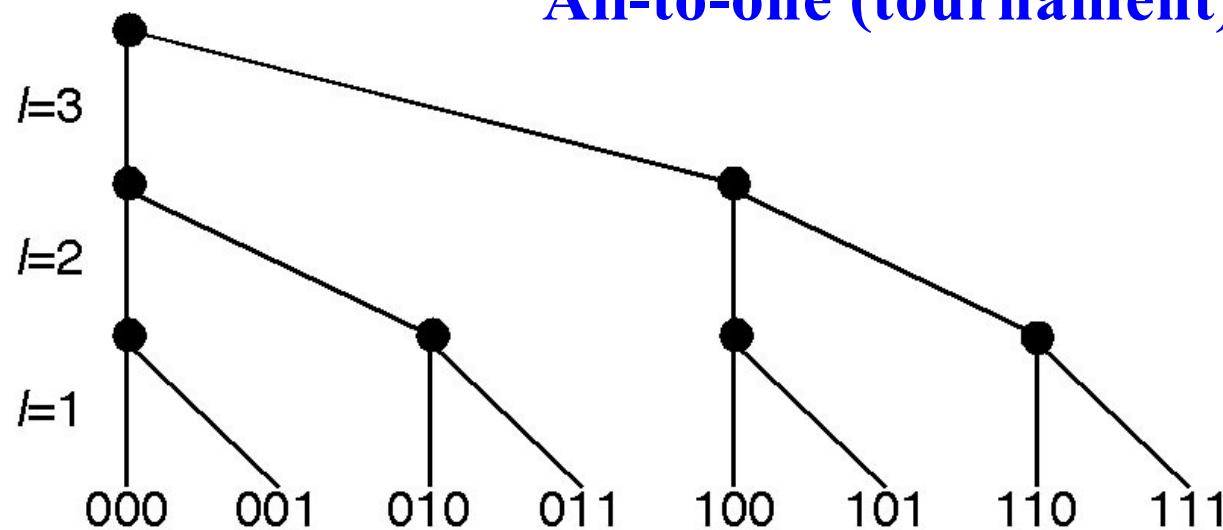
Scalable Global Communication

All-to-all (hypercube): $O(N \log N)$ $P < 10^3$



- Quicksort
- Fast Fourier transform
- All-to-all reduction

All-to-one (tournament): $O(N)$ $P > 10^5$



Exa-survivor

- Fast multipole method
- Multigrid method
- Wavelets
- All-to-one reduction

Multigrid Method

- **Residual equation:**

$$\frac{\mathbf{A}^{(l)}(\mathbf{v} + \mathbf{e}) = -4\pi e^2 \mathbf{n}}{\mathbf{A}^{(l)}\mathbf{v} = -4\pi e^2 \mathbf{n} + \mathbf{r}}$$

- **Smoothing**

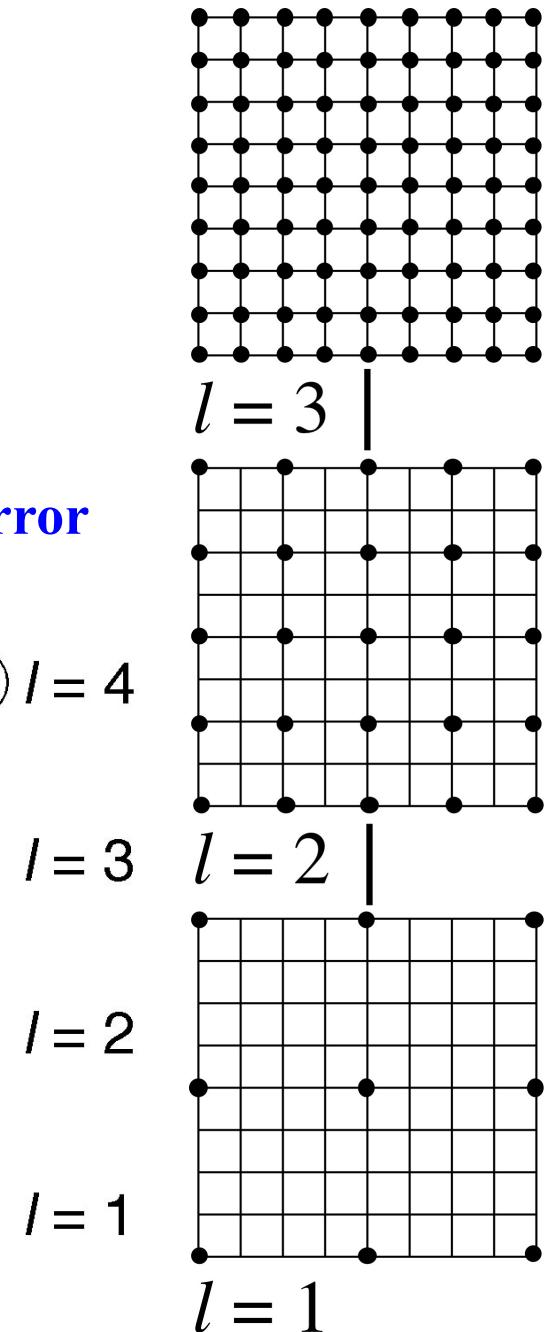
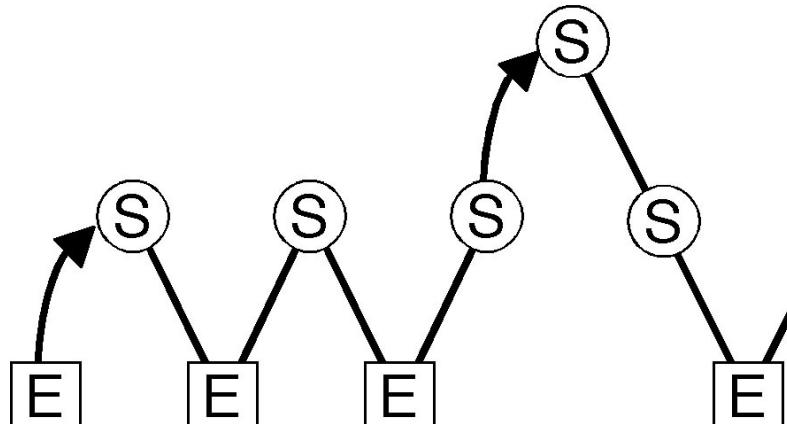
$$\mathbf{e} \leftarrow \left[1 + \mathbf{Z}^{(l)} \mathbf{A}^{(l)} \right] \mathbf{e} + \mathbf{Z}^{(l)} \mathbf{r}$$

- Coarsening (restriction) of residual & interpolation of error

$$\mathbf{r}^{(l-1)} \leftarrow \mathbf{Q}\mathbf{r}^{(l)}$$

$$\mathbf{e}^{(l)} \leftarrow \mathbf{P}\mathbf{e}^{(l-1)}$$

Solve partial differential equation by recursive coarsening



Adaptive Multigrids



Available online at www.sciencedirect.com



Computer Physics Communications 167 (2005) 151–164

Computer Physics
Communications

www.elsevier.com/locate/cpc

Embedded divide-and-conquer algorithm on hierarchical real-space grids: parallel molecular dynamics simulation based on linear-scaling density functional theory

Fuyuki Shimojo ^{a,b}, Rajiv K. Kalia ^a, Aiichiro Nakano ^{a,*}, Priya Vashishta ^a

^a Collaboratory for Advanced Computing and Simulations, Department of Computer Science, Department of Physics & Astronomy,
Department of Materials Science & Engineering, University of Southern California, Los Angeles, CA 90089-0242, USA

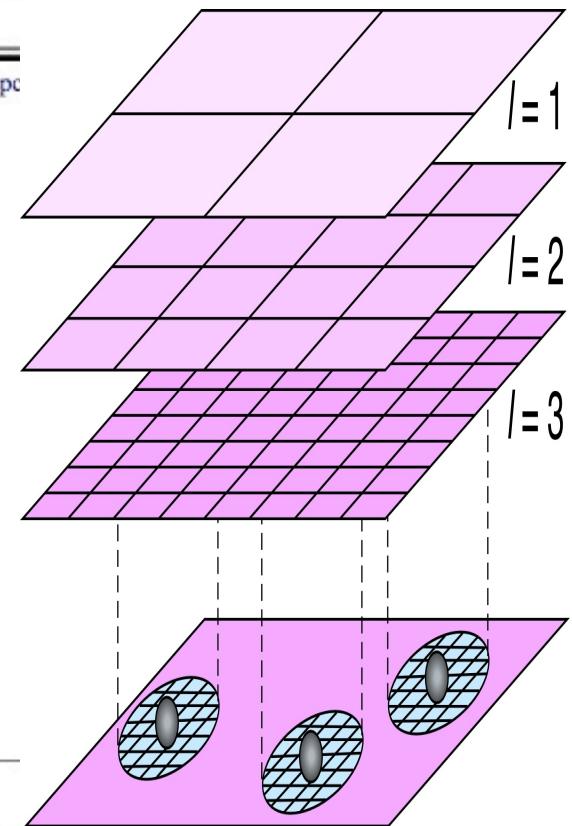
^b Department of Physics, Kumamoto University, Kumamoto 860-8555, Japan

Received 5 October 2004; received in revised form 24 January 2005; accepted 26 January 2005

Available online 14 March 2005

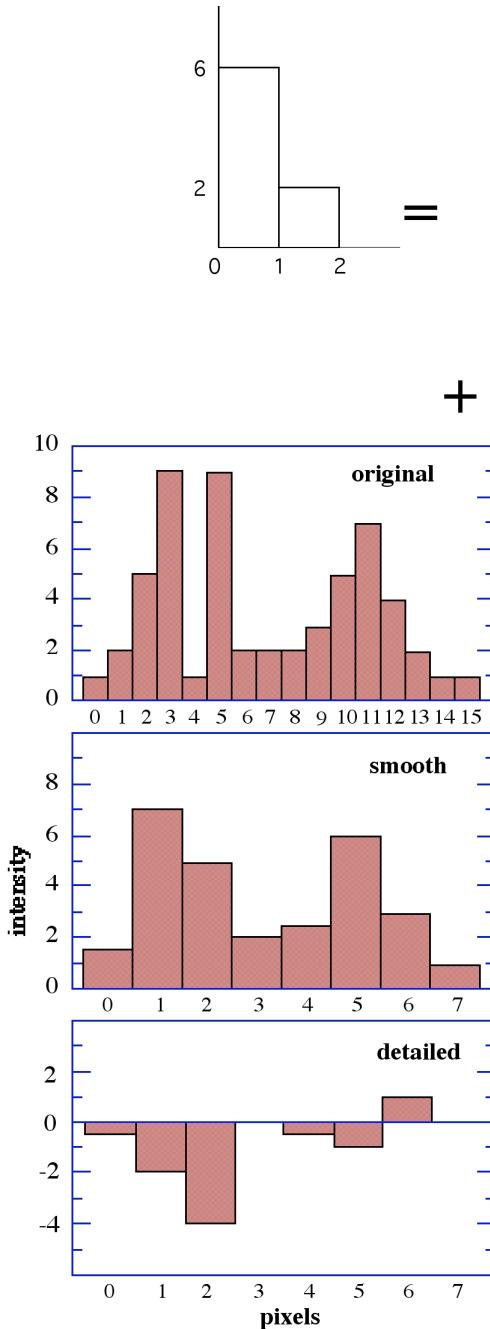
Abstract

A linear-scaling algorithm has been developed to perform large-scale molecular-dynamics (MD) simulations, in which interatomic forces are computed quantum mechanically in the framework of the density functional theory. A divide-and-conquer algorithm is used to compute the electronic structure, where non-additive contribution to the kinetic energy is included with an embedded cluster scheme. Electronic wave functions are represented on a real-space grid, which is augmented with coarse multigrids to accelerate the convergence of iterative solutions and adaptive fine grids around atoms to accurately calculate ionic pseudopotentials. Spatial decomposition is employed to implement the hierarchical-grid algorithm on massively parallel computers. A converged solution to the electronic-structure problem is obtained for a 32,768-atom amorphous CdSe system on 512 IBM POWER4 processors. The total energy is well conserved during MD simulations of liquid Rb, showing the applicability of this algorithm to first principles MD simulations. The parallel efficiency is 0.985 on 128 Intel Xeon processors for a 65,536-atom CdSe system.

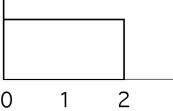


Also, Shimojo et al.,
Phys. Rev. B 77,
085103 ('08)

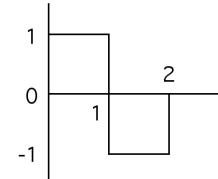
Wavelets



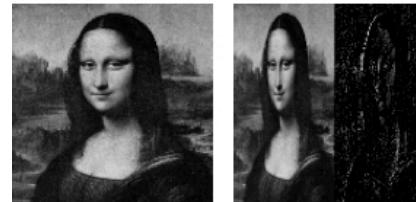
= $4 \times$



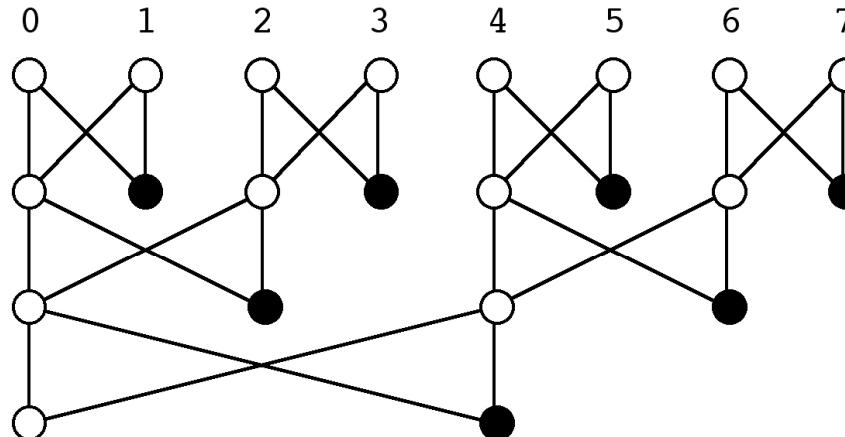
+ $2 \times$



transform rows



processors (pixels)



transform
columns



Used for image compression

○ smooth

● detail

E. J. Stollnitz, T. D. DeRose, & D. H. Salesin,
IEEE Computer Graphics Appl. 15(3), 76 ('95)

Wavelets for Model Reduction



Available online at www.sciencedirect.com



Finite Elements in Analysis and Design 43 (2007) 346–360

FINITE ELEMENTS
IN ANALYSIS
AND DESIGN

www.elsevier.com/locate/finel

Wavelet-based multi-scale coarse graining approach for DNA molecules

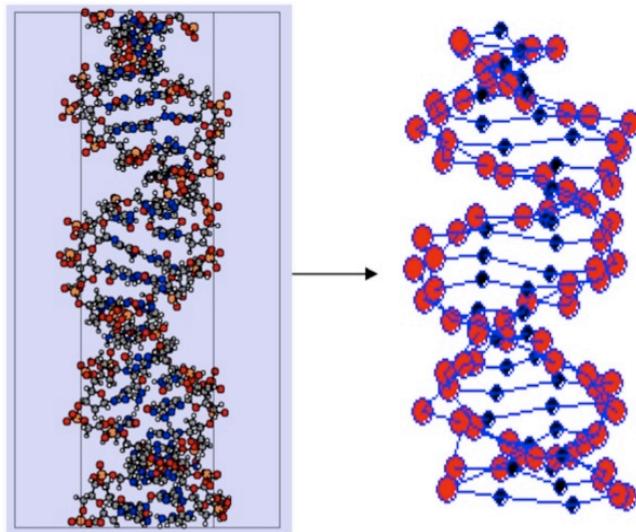
Jiun-Shyan Chen^{a,*}, Hailong Teng^a, Aiichiro Nakano^b

^aCivil and Environmental Engineering Department, University of California, Los Angeles, CA 90095-1593, USA

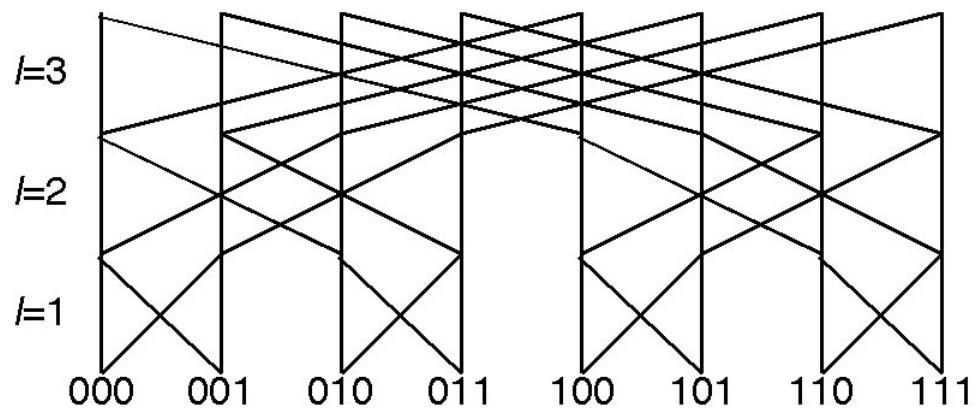
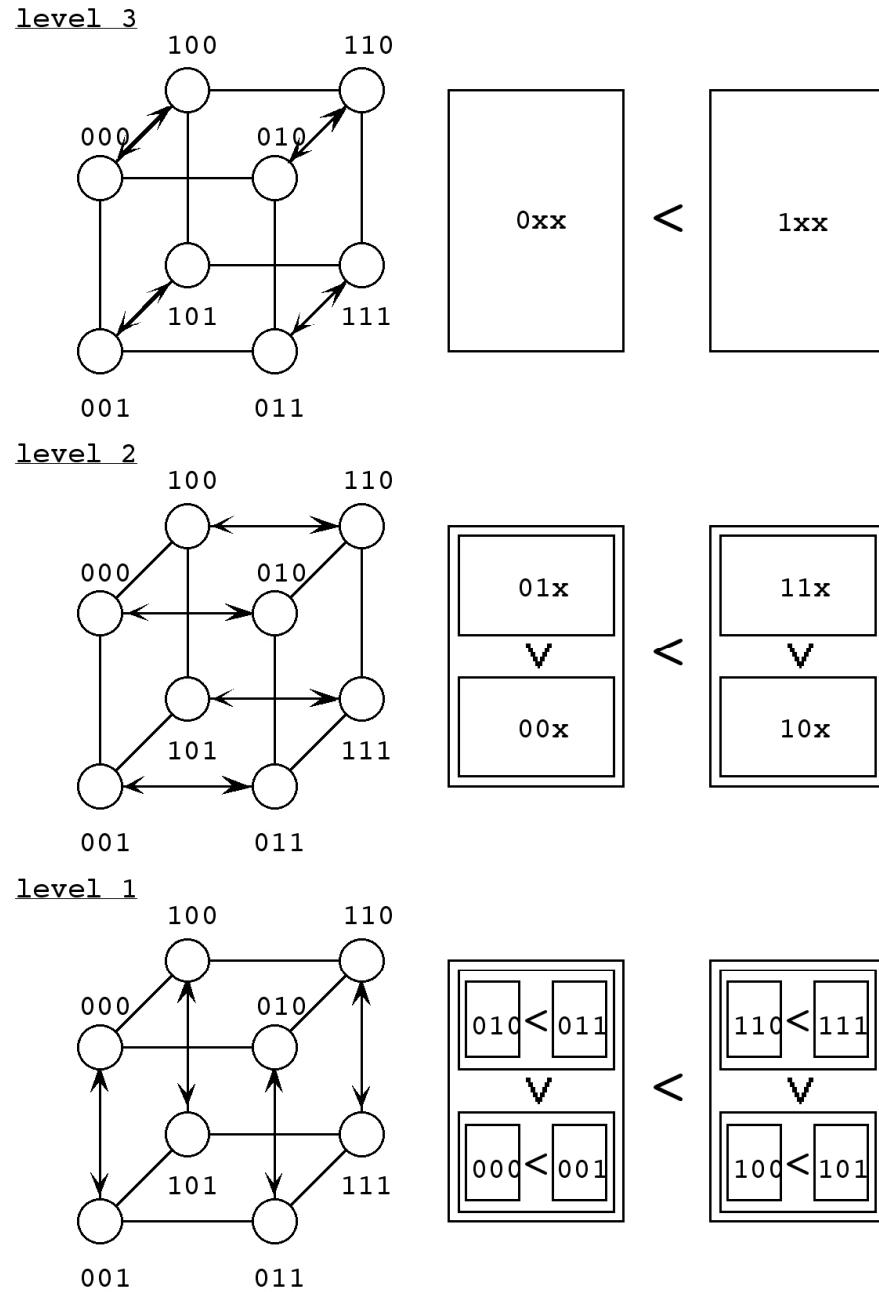
^bDepartment of Computer Science, University of Southern California, Los Angeles, CA 90089-0242, USA

Abstract

In this work, a coarse graining technique based on a multi-scale wavelet projection is proposed for modeling of DNA molecules. Based on the fine scale atomistic response and the Henderson's theorem, the distribution functions between centers of mass of two groups of atoms are employed to obtain the fine scale potential functions. These fine scale potential functions are then homogenized using the multi-scale wavelet projection to yield the coarse-scale effective potential functions between superatoms. Molecular dynamics simulation of DNA molecules under stretching process demonstrates that the results of fine scale model and the proposed coarse grained DNA model are in good agreement. The coarse grained simulation with a significant saving of computation time is shown to be in good agreement with the fine scale simulation. Due to the reduced spatial degrees of freedom and temporal discretization, a computational efficiency of three to four orders of magnitude is achieved in the proposed coarse grained model.



Hypercube Sort



Basis: Quicksort

In putting together this issue of *Computing in Science & Engineering*, we knew three things: it would be difficult to list just 10 algorithms; it would be fun to assemble the authors and read their papers; and, whatever we came up with in the end, it would be controversial. We tried to assemble the 10 algorithms with the greatest influence on the development and practice of science and engineering in the 20th century. Following is our list (here, the list is in chronological order; however, the articles appear in no particular order):

PHYS 516

- Metropolis Algorithm for Monte Carlo
- Simplex Method for Linear Programming
- Krylov Subspace Iteration Methods
- The Decompositional Approach to Matrix Computations
- The Fortran Optimizing Compiler
- QR Algorithm for Computing Eigenvalues
- Quicksort Algorithm for Sorting
- Fast Fourier Transform
- Integer Relation Detection
- Fast Multipole Method

CSCI 653

IEEE Comput. Sci. Eng. 2(1), 22 ('00)

A hybrid multi-loop genetic-algorithm/simplex/spatial-grid method for locating the optimum orientation of an adsorbed protein on a solid surface

Tao Wei ^a, Shengjing Mu ^b, Aiichiro Nakano ^c, Katherine Shing ^{a*}

^a Mork Family Department of Chemical Engineering and Material Science, University of Southern California, Los Angeles, CA, USA.

^b Yokogawa Engineering Asia Pte Ltd, Singapore.

^c Department of Computer Science, University of Southern California, Los Angeles, CA, USA.

Comput. Phys. Commun.
180, 669 ('09)

Quicksort: Divide-&-Conquer

```
quicksort(int list[],int left,int right) {  
    int j;  
    if (left < right) {  
        j = partition(list,left,right);  
        quicksort(list,left,j-1);  
        quicksort(list,j+1,right);  
    }  
}
```

- **partition:** Given `list[left:right]`, it first chooses the left-most element as a **pivot**; on return the **pivot** element is placed at the `j`-th position, &: i) `a[left],...,a[j-1]` are less than or equal to `a[j]`; ii) `a[j+1],...,a[right]` are greater than or equal to `a[j]`.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|-------|---|-------|---|-------|---|-------|-------|---|-------|
| [5 | 7 | 2 | 9 | 6 | 8 | 3 | 4 | 1 | 0] |
| [3 | 0 | 2 | 1 | 4] | 5 | [8 | 6 | 9 | 7] |
| [1 | 0 | 2] | 3 | [4] | 5 | [7 | 6] | 8 | [9] |
| [0] | 1 | [2] | 3 | [4] | 5 | [6] | 7 [] | 8 | [9] |

Sequential Quicksort

```
void quicksort(int list[],int left,int right) {  
    int pivot,i,j;  
    int temp;  
  
    if (left < right) {  
        i = left; j = right + 1;  
        pivot = list[left];  
        do {  
            while (list[++i] < pivot);  
            while (list[--j] > pivot);  
            if (i < j) {  
                temp = list[i]; list[i] = list[j]; list[j] = temp;  
            }  
        } while (i < j);  
        temp = list[left]; list[left] = list[j]; list[j] = temp;  
        quicksort(list,left,j-1);  
        quicksort(list,j+1,right);  
    }  
}
```

Parallel Quicksort

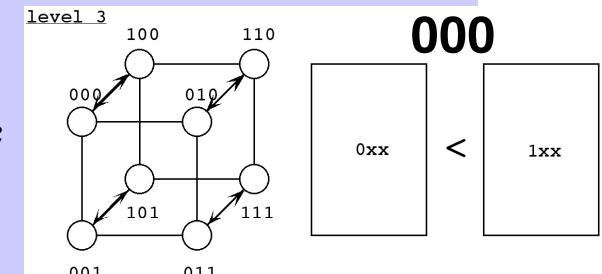
```

bitvalue := 2dimension-1; 100
mask := 2dimension - 1; 111
for L := dimension downto 1
begin
    if myid AND mask = 0 then
        choose a pivot value for the L-dimensional subcube; pivot = avg(list elements)
        broadcast the pivot from the master to the other members of the subcube;
        partition list[0:nelement-1] into two sublists such that
        list[0:j] ≤ pivot < list[j+1:nelement-1];
        partner := myid XOR bitvalue;
        if myid AND bitvalue = 0 then
            begin
                send the right sublist list[j+1:nelement-1] to partner;
                receive the left sublist of partner;
                append the received list to my left list
            end
        else
            begin
                send the left sublist list[0:j] to partner;
                receive the right sublist of partner;
                append the received list to my right list
            end
        nelement := nelement - nsend + nreceive;
        mask = mask XOR bitvalue;
        bitvalue = bitvalue/2;
    end
sequential quicksort to list[0:nelement-1]

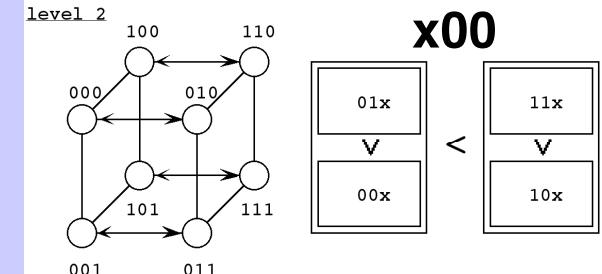
```

Subcube master:

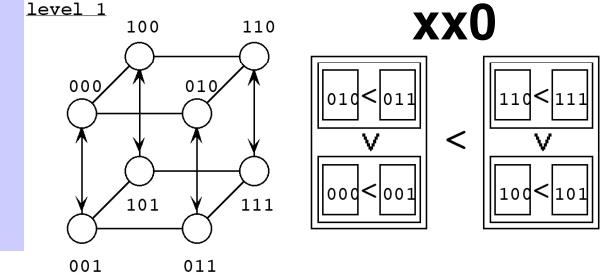
000



x00



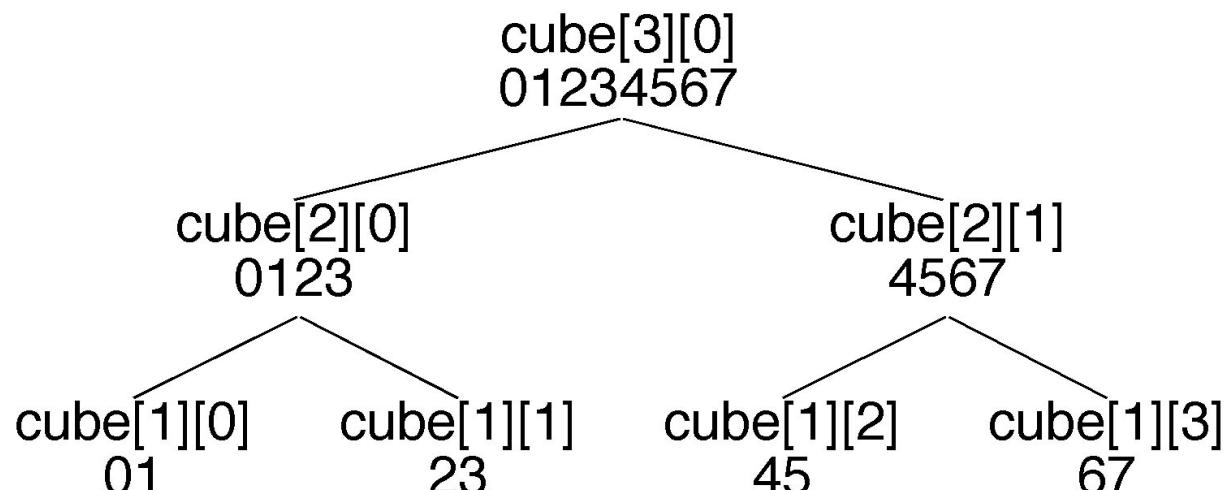
xx0



Subcube Broadcast

```
bitvalue = nprocs >> 1;
mask = nprocs - 1;

for (L=dimension; L>=1; L--) {
    ...
    if ((myid & mask) == 0)
        Calculate the pivot as the average of the local list
        element values
    MPI_Bcast(&pivot,1,MPI_INT,0,cube[L][myid/nprocs_cube]);
    ...
    mask = mask ^ bitvalue; /* Flip the current bit to 0 */
    bitvalue = bitvalue >> 1; /* Next significant bit */
}
```



MPI Communicators

Recursive bisection of processor groups

```
MPI_Comm cube[MAXD][MAXP];
MPI_Group cube_group[MAXD][MAXP];
...
MPI_Comm_size(MPI_COMM_WORLD,&nprocs_cube);
cube[3][0] = MPI_COMM_WORLD;
...
// At each level
MPI_Comm_group(cube[L][c],&(cube_group[L][c]));
nprocs_cube = nprocs_cube/2;
for(p=0; p<nprocs_cube; p++) procs_cube[p] = p;
MPI_Group_incl(cube_group[L][c],nprocs_cube,procs_cube,&(cube_group[L-1][2*c]));
MPI_Group_excl(cube_group[L][c],nprocs_cube,procs_cube,&(cube_group[L-1][2*c+1]));
MPI_Comm_create(cube[L][c],cube_group[L-1][2*c],&(cube[L-1][2*c]));
MPI_Comm_create(cube[L][c],cube_group[L-1][2*c+1],&(cube[L-1][2*c+1]));
MPI_Group_free(&(cube_group[L][c]));
MPI_Group_free(&(cube_group[L-1][2*c]));
MPI_Group_free(&(cube_group[L-1][2*c+1]));
```

$$c = \text{myid}/\text{nprocs_cube}$$

Hypercube Topology

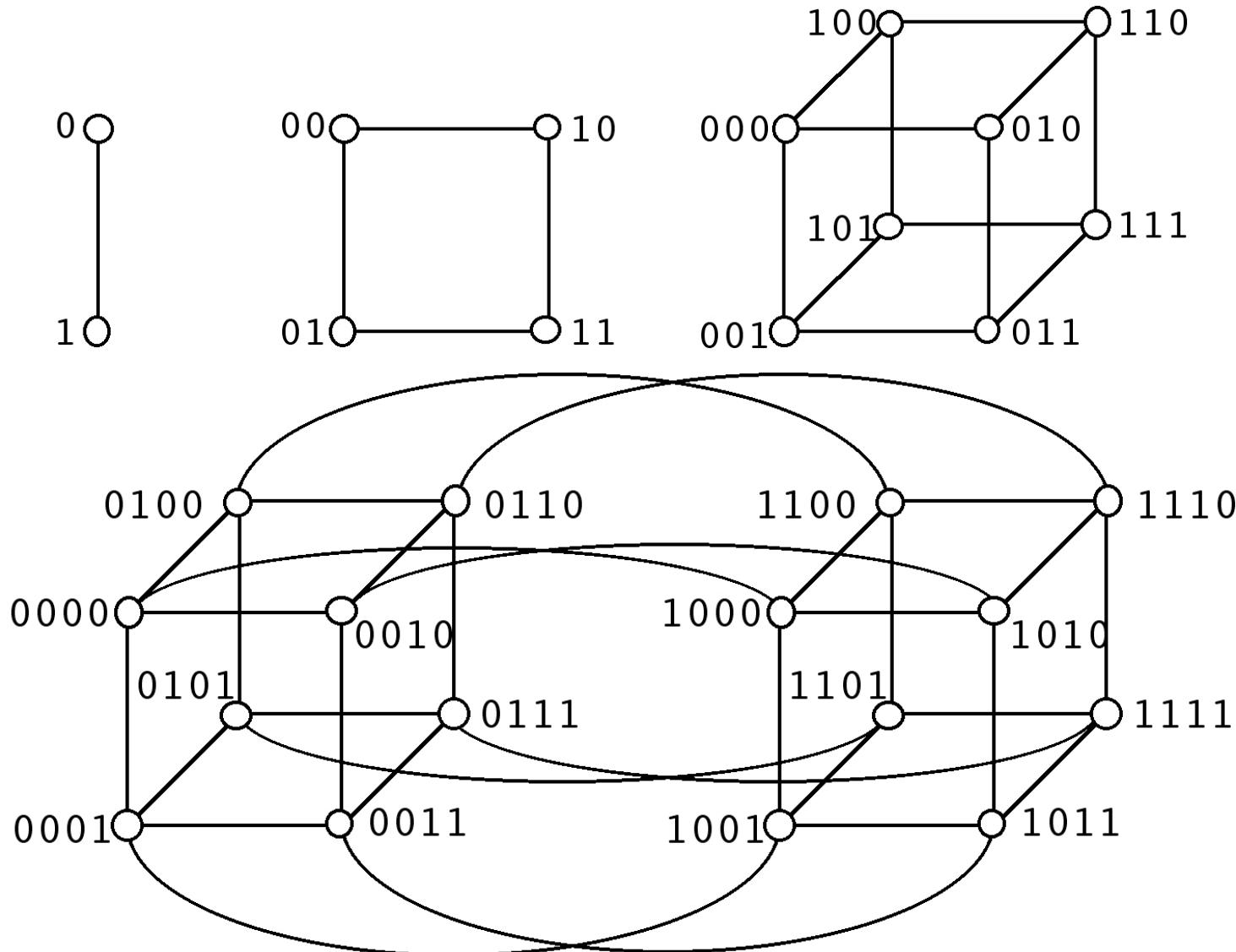
- **Hamming distance:** The total number of bit positions at which two binary numbers differ.
- In a hypercube network topology, two nodes are connected if their Hamming distance is 1. The *connectivity* of a d -dimensional (or n -node) hypercube is thus d . Since each link can change only one digit, the *diameter* is d or $\log_2 n$.

Recursive hypercube construction algorithm

- (1) A one-dim. hypercube has two connected nodes 0 & 1.
- (2) A $(d+1)$ -dim. hypercube is defined from a d -dim. hypercube:
 - a. Duplicate the d -dim. hypercube including node numbers.
 - b. Create links between nodes with the same number in the original & duplicate.
 - c. Append a binary 1 to the left of each node number in the duplicate, & a binary 0 to left of each node number in the original.

See [Gramma](#), Secs. 2.4.2-2.4.4

Hypercube



- $\log_2 n$ network interfaces required per node for an n -node parallel computer
- Hypercube algorithms only use direct network links

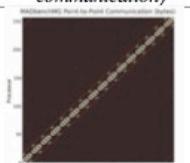
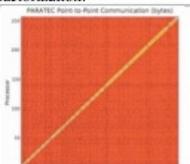
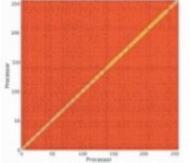
Analysis of Parallel Quicksort

```
for L := dimension downto 1 logp
begin
  if master then
    choose a pivot value for the L-dimensional subcube; n/p
    broadcast the pivot from the master to the subcube members; [1,logp]
    partition list[0:nelement-1] into two sublists such that n/p
    list[0:j] ≤ pivot < list[j+1:nelement-1];
    if lower_partner then
      begin
        send the right sublist list[j+1:nelement-1] to partner;
        receive the left sublist of partner; n/p
      end
    else if higher_partner then
      begin
        send the left sublist list[0:j] to partner;
        receive the right sublist of partner; n/p
      end
    nelement := nelement - nsend + nreceive;
  end
  sequential quicksort to list[0:nelement-1] (n/p)log(n/p)
```

$$T_{\text{average}} = O\left(\frac{n}{p} \log \frac{n}{p}\right) + O\left(\frac{n}{p} \log p\right) + O\left(\log^2 p\right)$$

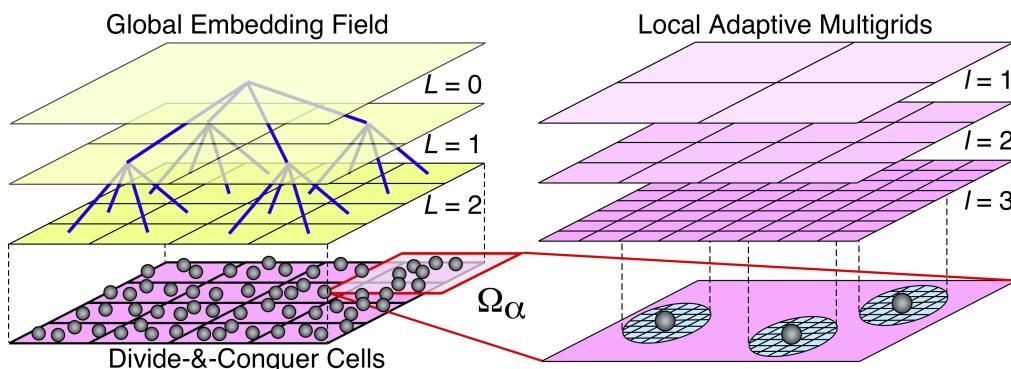
The Landscape of Parallel Computing Research: A View from Berkeley

7 dwarfs (dwarf = algorithmic method that captures a pattern of computation & communication) + 6 combinatorial dwarfs

| Dwarf | Description | Communication Pattern (Figure axes show processors 1 to 256, with black meaning no communication) | NAS Benchmark / Example HW | Dwarf | Description | Communication Pattern (Figure axes show processors 1 to 256, with black meaning no communication) | NAS Benchmark / Example HW |
|--|---|--|---|--|--|--|---|
| 1. Dense Linear Algebra (e.g., BLAS [Blackford et al 2002], ScALAPACK [Blackford et al 1996], or MATLAB [MathWorks 2006]) | Data are dense matrices or vectors. (BLAS Level 1 = vector-vector; Level 2 = matrix-vector; and Level 3 = matrix-matrix.) Generally, such applications use unit-stride memory accesses to read data from rows, and strided accesses to read data from columns. |  | Block Triadiagonal Matrix, Lower Upper Symmetric Gauss-Seidel / Vector computers, Array computers | 4. N-Body Methods (e.g., Barnes-Hut [Barnes and Hut 1986], Fast Multipole Method [Greengard and Rokhlin 1987]) | Depends on interactions between many discrete points. Variations include particle-particle methods, where every point depends on all others, leading to an $O(N^2)$ calculation, and hierarchical particle methods, which combine forces or potentials from multiple points to reduce the computational complexity to $O(N \log N)$ or $O(N)$. |  | (no benchmark) / GRAPE [Tokyo 2006], MD-GRAPE [IBM 2006] |
| 2. Sparse Linear Algebra (e.g., SpMV, OSKI [OSKI 2006], or SuperLU [Dennmell et al 1999]) | Data sets include many zero values. Data is usually stored in compressed matrices to reduce the storage and bandwidth requirements to access all of the nonzero values. One example is block compressed sparse row (BCSR). Because of the compressed formats, data is generally accessed with indexed loads and stores. |  | Conjugate Gradient / Vector computers with gather/scatter | 5. Structured Grids (e.g., Cactus [Goodale et al 2003] or Lattice-Boltzmann Magneto-hydrodynamics [LBMHD 2005]) | Represented by a regular grid; points on grid are conceptually updated together. It has high spatial locality. Updates may be in place or between 2 versions of the grid. The grid may be subdivided into finer grids in areas of interest ("Adaptive Mesh Refinement"); and the transition between granularities may happen dynamically. |  | Multi-Grid, Scalar Penta-diagonal / QCDOC [Edinburg 2006], BlueGeneL |
| 3. Spectral Methods (e.g., FFT [Cooley and Tukey 1965]) | Data are in the frequency domain, as opposed to time or spatial domains. Typically, spectral methods use multiple butterfly stages, which combine multiply-add operations and a specific pattern of data permutation, with all-to-all communication for some stages and strictly local for others. |  | Fourier Transform / DSPs, Zalink PDSP [Zurlink 2006] | 6. Unstructured Grids (e.g., ABAQUS [ABAQUS 2006] or FIDAP [FLUENT 2006]) | An irregular grid where data locations are selected, usually by underlying characteristics of the application. Data point location and connectivity of neighboring points must be explicit. The points on the grid are conceptually updated together. Updates typically involve multiple levels of memory reference indirection, as an update to any point requires first determining a list of neighboring points, and then loading values from those neighboring points. |  | Unstructured Adaptive / Vector computers with gather/scatter, Tera Multi Threaded Architecture [Berry et al 2006] |
| | |  | | 7. Monte Carlo (e.g., Quantum Monte Carlo [Aspuru-Guzik et al 2005]) | Calculations depend on statistical results of repeated random trials. Considered embarrassingly parallel. |  | Embarrassingly Parallel / NSF Teragrid |

A Metascalable Dwarf

- A metascalable (or “design once, scale on new architectures”) parallel computing framework: Reduce diverse applications (e.g., equation solvers, constrained optimization, search, visualization & graphs) to a scalable form using divide-conquer-recombine (DCR)



F. Shimojo et al., *J. Phys.: Condens. Matter* 20, 294204 ('08); *Int'l J. High Perf. Comput. Appl.* 22, 113 ('08); *IPDPS09* (IEEE); *SC14* (IEEE/ACM); *Comput. Sci. & Eng.* 21, 64 ('19); *HPCAsia20* (IEEE)

- The DCR framework encompasses all 7 dwarfs (algorithmic methods that capture computation & communication patterns):

- (1) dense linear algebra; (2) sparse linear algebra; (3) spectral methods;
- (4) N-body methods;
- (5) structured grids;
- (6) unstructured grids;
- (7) Monte Carlo

later expanded with
6 combinatorial dwarfs

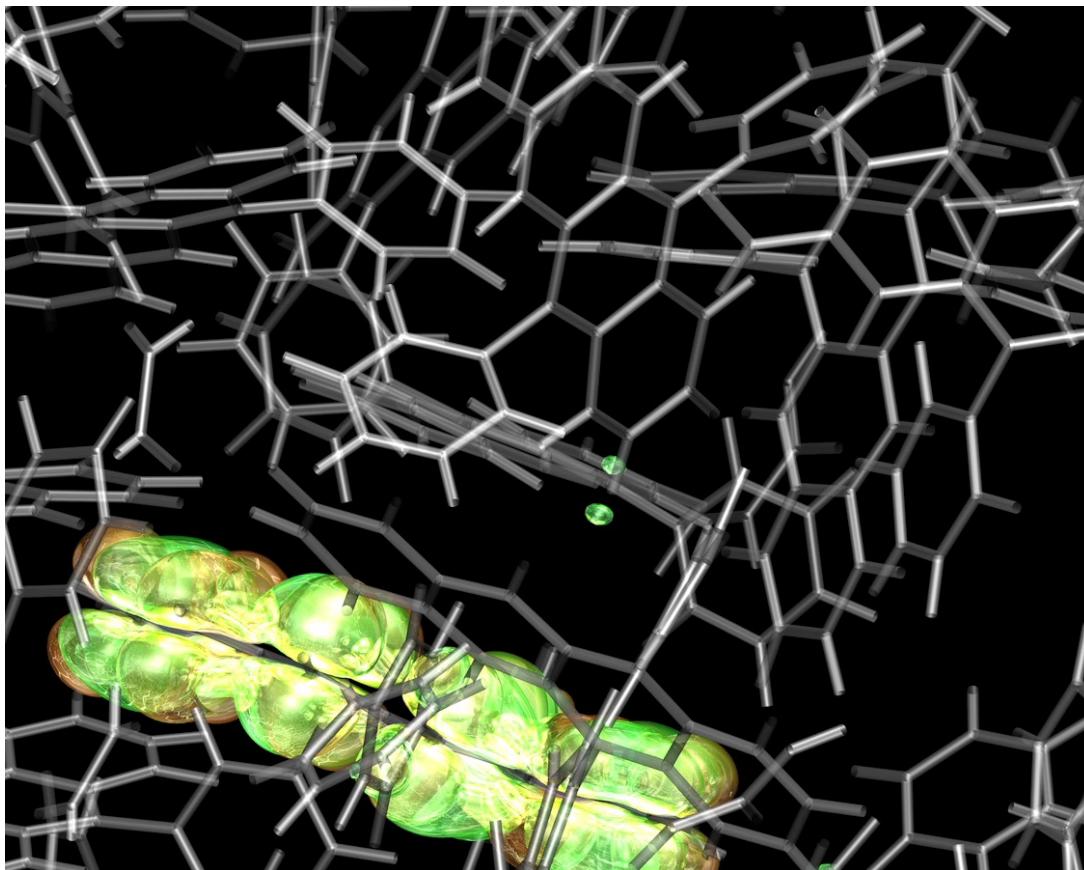
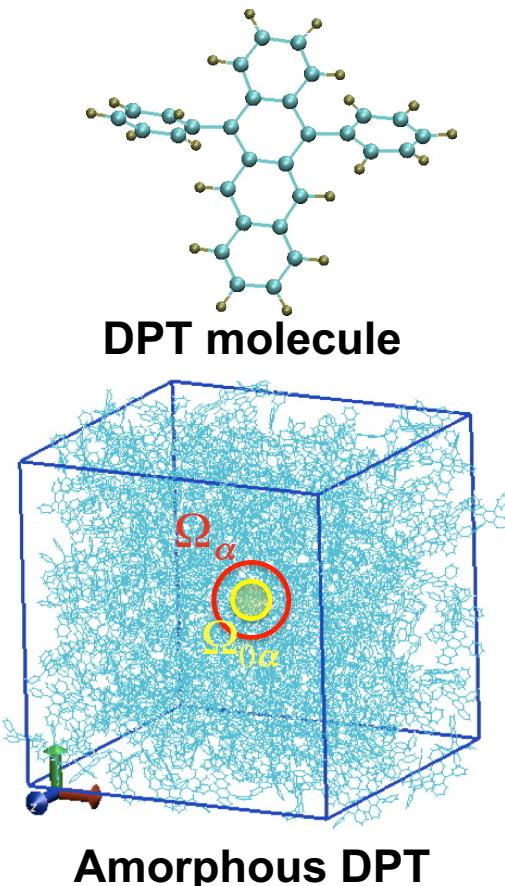


Dr. David Patterson
Professor,
UC Berkeley

The Parallel Revolution has Started: Are You Part of the Solution or Part of the Problem?

Singlet Fission in Amorphous DPT

- Move up from molecules to microstructures
- Challenge: Unprecedented 10^4 -atom NAQMD simulation
- Computational approach: Divide-conquer-recombine (DCR) NAQMD

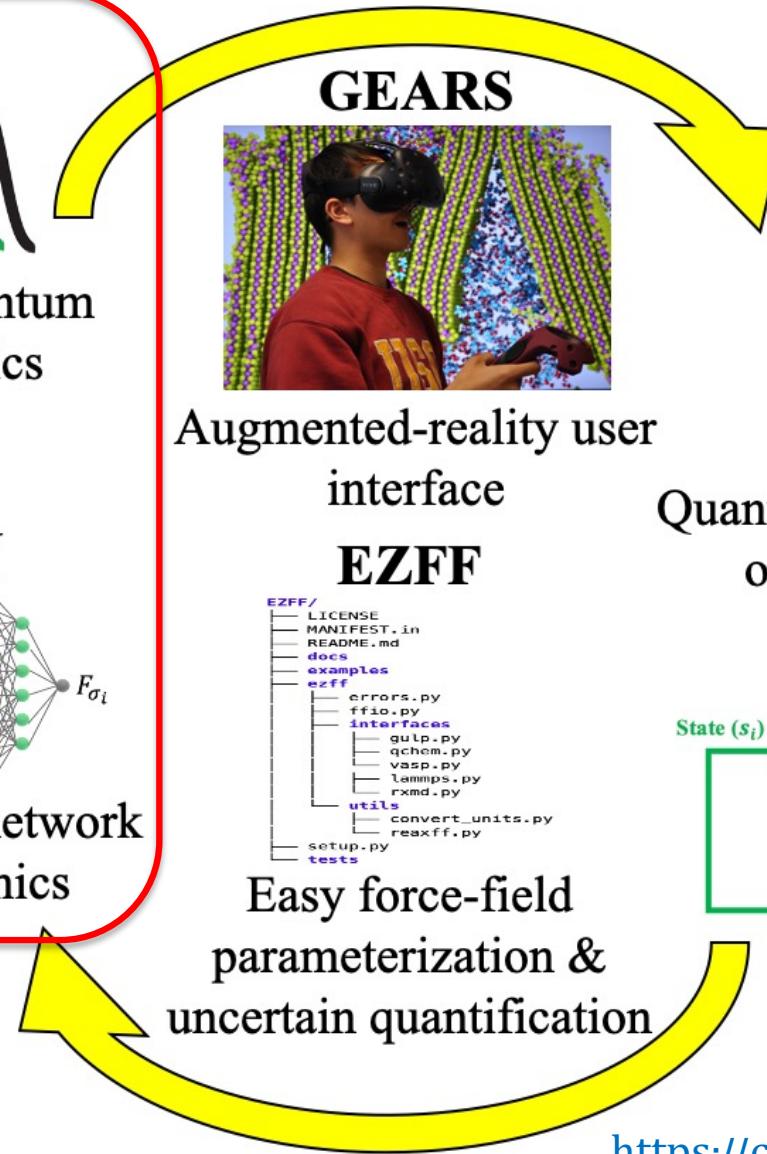
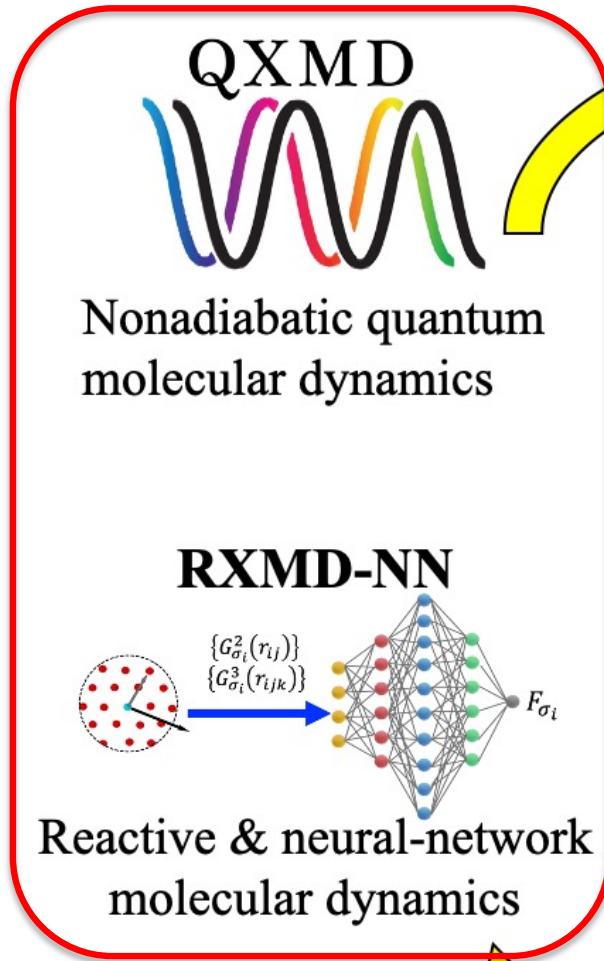


NAQMD:
nonadiabatic
quantum
molecular
dynamics

- Divide-conquer-recombine NAQMD (phonon-assisted exciton dynamics) + time-dependent perturbation theory (singlet-fission rate) + kinetic Monte Carlo calculations of exciton population dynamics in 6,400-atom amorphous DPT

DCR Engine in AIQ-XMaS Software Suite

AI & Quantum-Computing Enabled Exa-Materials Simulator



Easy force-field parameterization & uncertain quantification

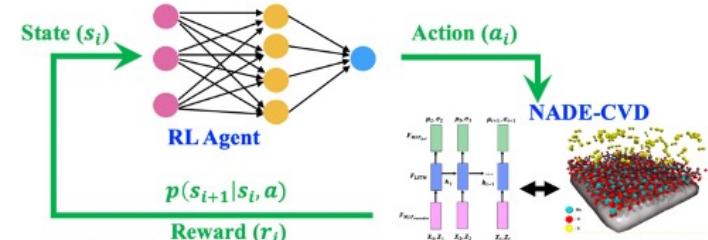
DCR engine

MISTIQS



Quantum many-body dynamics on quantum computers

MAITAI



AI tools for materials design

<https://cybermagics.netlify.app/software>