

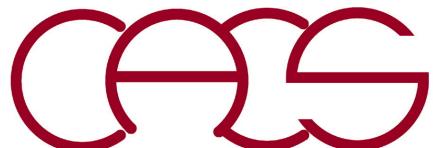
Quantum Dynamics

Aiichiro Nakano

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

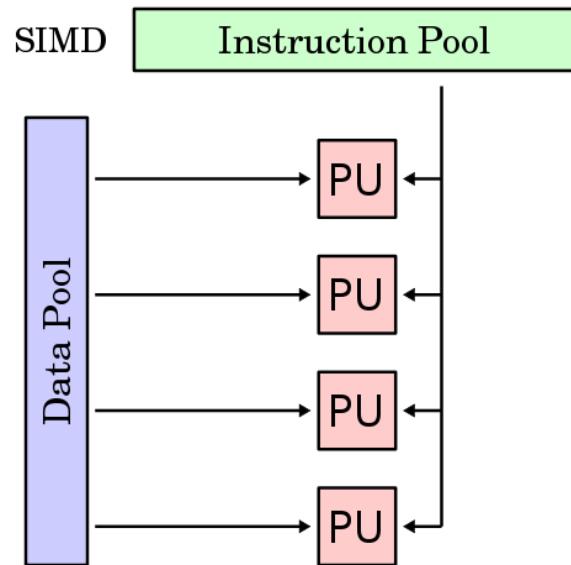
Email: anakano@usc.edu

Objective: Learn prototypical continuum simulation



What to Learn with QD?

- An archetype of single-instruction multiple-data (SIMD) parallel applications: Continuum simulations based on partial differential equation (PDE)
- Alternative parallelization methods with distinct scalability
- Performance optimization on emerging accelerators



SIMD: A single instruction operates on multiple different data streams

Complementary to molecular dynamics (MD) = multiple-instruction multiple-data (MIMD), discrete, ordinary differential equation (ODE)

Wave Equation

- Complex wave function

$$\psi(x,t) = \text{Re}\psi(x,t) + i\text{Im}\psi(x,t) \in \mathbf{C} \quad (i = \sqrt{-1})$$

- Normalization

$$\int dx |\psi(x,t)|^2 = 1$$

- Schrödinger equation (in atomic unit)

$$i \frac{\partial}{\partial t} \psi(x,t) = H \psi(x,t)$$

- Hamiltonian operator

$$H = -\frac{1}{2} \frac{\partial^2}{\partial x^2} + V(x) = T_x + V$$

- Periodic boundary condition

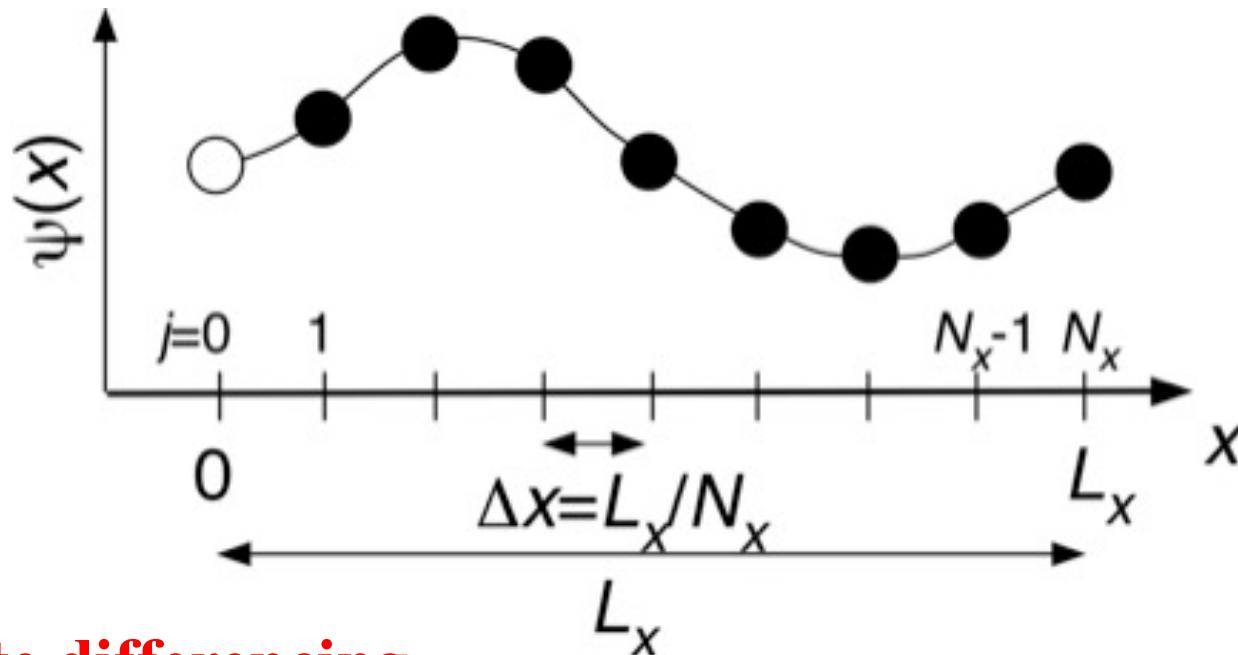
$$\psi(x + L_x) = \psi(x)$$



Erwin Schrödinger
1933 Nobel Physics
prize

Spatial Discretization

- Regular 1D mesh: $\psi_j = \psi(j\Delta x)$ ($\Delta x = L_x/N_x$)



- Finite differencing

$$\begin{cases} (T_x \psi)_j = -\frac{1}{2} \frac{\psi_{j-1} - 2\psi_j + \psi_{j+1}}{(\Delta x)^2} \\ (V \psi)_j = V_j \psi_j \end{cases}$$

$$\begin{aligned} \frac{\partial^2}{\partial x^2} \psi(x, t) &= \frac{\partial \psi(x+\Delta/2, t)/\partial x - \partial \psi(x-\Delta/2, t)/\partial x}{\Delta} \\ &= \frac{\frac{\psi(x+\Delta, t) - \psi(x, t)}{\Delta} - \frac{\psi(x, t) - \psi(x-\Delta, t)}{\Delta}}{\Delta} \\ &= \frac{\psi(x+\Delta, t) - 2\psi(x, t) + \psi(x-\Delta, t)}{\Delta^2} \end{aligned}$$

Temporal Propagation

- **Formal solution to the Schrödinger equation:** $\frac{\partial}{\partial t}\psi(t) = -iH\psi(t)$

$$\psi(t + \Delta t) = \exp(-iH\Delta t)\psi(t) \quad \exp(-i\hat{H}t) = \sum_{n=0}^{\infty} \frac{(-it)^n}{n!} \hat{H}^n$$

- **Split-operator method: unitary!**

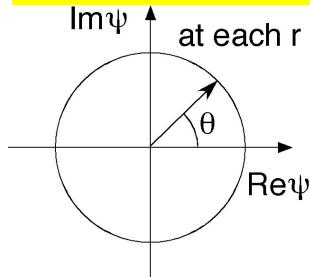
$$\begin{aligned} \psi(t + \Delta t) &= \exp(-i(T_x + V)\Delta t)\psi(t) \\ &= \exp(-iV\Delta t/2)\exp(-iT_x\Delta t)\exp(-iV\Delta t/2)\psi(t) + O([\Delta t]^3) \end{aligned}$$

Split in a way each operator is easily exponentiated

- **Potential propagator (mesh point-by-point complex-number multiplications)**

$$(u_0 + iu_1)(\psi_0 + i\psi_1) = (u_0\psi_0 - u_1\psi_1) + i(u_1\psi_0 + u_0\psi_1)$$

$$(\exp(-iV\Delta t/2)\psi)_j = \exp(-iV_j\Delta t/2)\psi_j$$



Rotation

$$\begin{aligned} &= [\cos(V_j\Delta t/2) - i \sin(V_j\Delta t/2)][\text{Re}\psi_j + i \text{Im}\psi_j] \\ &= [\cos(V_j\Delta t/2)\text{Re}\psi_j + \sin(V_j\Delta t/2)\text{Im}\psi_j] \\ &\quad + i[-\sin(V_j\Delta t/2)\text{Re}\psi_j + \cos(V_j\Delta t/2)\text{Im}\psi_j] \end{aligned}$$

$$\begin{aligned} \exp(ia) &= \sum_{n=0}^{\infty} \frac{(ia)^n}{n!} = \left(1 - \frac{a^2}{2!} + \frac{a^4}{4!} - \dots\right) + i\left(a - \frac{a^3}{3!} + \frac{a^5}{5!} - \dots\right) \\ &= \cos(a) + i\sin(a) \quad \begin{bmatrix} \cos(V_{jk}\Delta t/2) & \sin(V_{jk}\Delta t/2) \\ -\sin(V_{jk}\Delta t/2) & \cos(V_{jk}\Delta t/2) \end{bmatrix} \begin{bmatrix} \text{Re}\psi_{jk} \\ \text{Im}\psi_{jk} \end{bmatrix} \end{aligned}$$

Kinetic Propagator: It's a Matrix!

- Mesh-point coupling

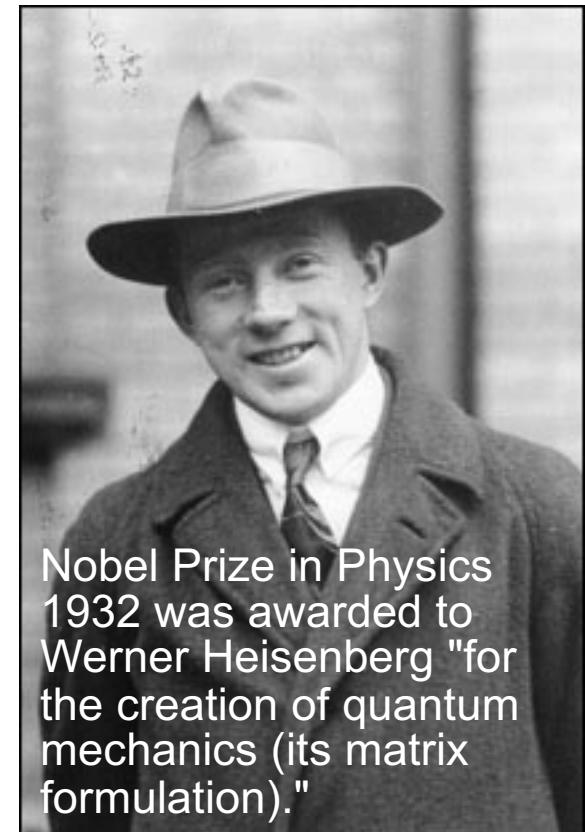
$$T_x \psi_j = b\psi_{j-1} + 2a\psi_j + b\psi_{j+1}$$

- Tridiagonal matrix representation

$$T_x = \begin{bmatrix} 2a & b & & & & & b \\ b & 2a & b & & & & \\ & b & 2a & b & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & b & 2a & b & \\ & & & & b & 2a & b \\ & & & & & b & 2a \end{bmatrix}$$

Note the periodic boundary condition

$$\begin{cases} a = 1/2(\Delta x)^2 \\ b = -1/2(\Delta x)^2 \end{cases}$$



Nobel Prize in Physics 1932 was awarded to Werner Heisenberg "for the creation of quantum mechanics (its matrix formulation)."



The Matrix (1999)

Space Splitting Method (SSM)

- **2×2 block-diagonal decomposition & split-operator exponentiation**

$$T_x = \begin{bmatrix} 2a & b & & b \\ b & 2a & b & \\ & b & 2a & b \\ & & \ddots & \ddots & \ddots \\ & & b & 2a & b \\ & & & b & 2a & b \\ & & & & b & 2a \\ b & & & & b & 2a \end{bmatrix}$$

Block-by-block exponentiation

$$\exp \left[\begin{bmatrix} \blacksquare & & \\ & \blacksquare & \\ & & \blacksquare \end{bmatrix} \right] = \sum_{n=0}^{\infty} \frac{1}{n!} \left[\begin{bmatrix} \blacksquare^n & & \\ & \blacksquare^n & \\ & & \blacksquare^n \end{bmatrix} \right]^n = \begin{bmatrix} e^{\blacksquare} & & \\ & e^{\blacksquare} & \\ & & e^{\blacksquare} \end{bmatrix}$$

Split-operator (Trotter expansion) again

$$= \frac{1}{2} \left[\begin{bmatrix} a & b & & & b \\ b & a & & & \\ & a & b & & \\ & b & a & & \\ & & \ddots & & \\ & & a & b & \\ & & b & a & \end{bmatrix} + \begin{bmatrix} a & & & & b \\ & a & b & & \\ & b & a & & \\ & & \ddots & & \\ & & a & b & \\ & & b & a & \\ & & & & a \end{bmatrix} \right] + \frac{1}{2} \left[\begin{bmatrix} a & b & & & b \\ b & a & & & \\ & a & b & & \\ & b & a & & \\ & & \ddots & & \\ & & a & b & \\ & & b & a & \end{bmatrix} \right]$$

$$\exp(-i\Delta t T_x) = U_x^{(\text{half})} U_x^{(\text{full})} U_x^{(\text{half})} + \mathcal{O}([\Delta t]^3)$$

$$= \exp \left(-\frac{i\Delta t}{2} \left[\begin{bmatrix} a & b & & & b \\ b & a & & & \\ & a & b & & \\ & b & a & & \\ & & \ddots & & \\ & & a & b & \\ & & b & a & \end{bmatrix} \right] \right) \exp \left(-i\Delta t \left[\begin{bmatrix} a & & & & b \\ & a & b & & \\ & b & a & & \\ & & \ddots & & \\ & & a & b & \\ & & b & a & \\ & & & & a \end{bmatrix} \right] \right) \exp \left(-\frac{i\Delta t}{2} \left[\begin{bmatrix} a & b & & & b \\ b & a & & & \\ & a & b & & \\ & b & a & & \\ & & \ddots & & \\ & & a & b & \\ & & b & a & \end{bmatrix} \right] \right)$$

How? Block diagonal \rightarrow block-by-block exponentiation

Space Splitting Method (SSM)

$$\begin{aligned}
 &= \exp\left(-\frac{i\Delta t}{2}\begin{bmatrix} a & b \\ b & a \\ & \ddots \\ & & a & b \\ & & b & a \\ & & & \ddots \\ & & & & a & b \\ & & & & b & a \end{bmatrix}\right) \exp\left(-i\Delta t\begin{bmatrix} a & & & b \\ & a & b & \\ & b & a & \\ & & \ddots & \\ & & & a & b \\ & & & b & a \\ & & & & a \end{bmatrix}\right) \exp\left(-\frac{i\Delta t}{2}\begin{bmatrix} a & b \\ b & a \\ & \ddots \\ & & a & b \\ & & b & a \\ & & & \ddots \\ & & & & a & b \\ & & & & b & a \end{bmatrix}\right) \\
 &= \begin{bmatrix} \varepsilon_2^+ & \varepsilon_2^- \\ \varepsilon_2^- & \varepsilon_2^+ \\ & \ddots \\ & & \varepsilon_2^+ & \varepsilon_2^- \\ & & \varepsilon_2^- & \varepsilon_2^+ \\ & & & \ddots \\ & & & & \varepsilon_2^+ & \varepsilon_2^- \\ & & & & \varepsilon_2^- & \varepsilon_2^+ \end{bmatrix} \begin{bmatrix} \varepsilon_1^+ & & & & & & \\ & \varepsilon_1^+ & \varepsilon_1^- & & & & \\ & \varepsilon_1^- & \varepsilon_1^+ & \ddots & & & \\ & & & \ddots & \varepsilon_1^+ & \varepsilon_1^- \\ & & & & \varepsilon_1^- & \varepsilon_1^+ \\ & & & & & \ddots \\ & & & & & & \varepsilon_1^+ \end{bmatrix} \begin{bmatrix} \varepsilon_2^+ & \varepsilon_2^- \\ \varepsilon_2^- & \varepsilon_2^+ \\ & \ddots \\ & & \varepsilon_2^+ & \varepsilon_2^- \\ & & \varepsilon_2^- & \varepsilon_2^+ \\ & & & \ddots \\ & & & & \varepsilon_2^+ & \varepsilon_2^- \\ & & & & \varepsilon_2^- & \varepsilon_2^+ \end{bmatrix}
 \end{aligned}$$

$$\begin{cases} \varepsilon_n^+ = \frac{1}{2} \left[\exp\left(-\frac{i\Delta t}{n}(a+b)\right) + \exp\left(-\frac{i\Delta t}{n}(a-b)\right) \right] \\ \varepsilon_n^- = \frac{1}{2} \left[\exp\left(-\frac{i\Delta t}{n}(a+b)\right) - \exp\left(-\frac{i\Delta t}{n}(a-b)\right) \right] \end{cases} \quad \text{Just need } 2 \times 2 \text{ exponentiation} \quad \exp\left(-\frac{i\Delta t}{(2)} \begin{bmatrix} a & b \\ b & a \end{bmatrix}\right)$$

Use eigen-decomposition & telescoping

$$(UDU^{-1})^n = \overbrace{UDU^{-1} \ UDU^{-1} \ \dots \ UDU^{-1}}^n = UD^nU^{-1}$$

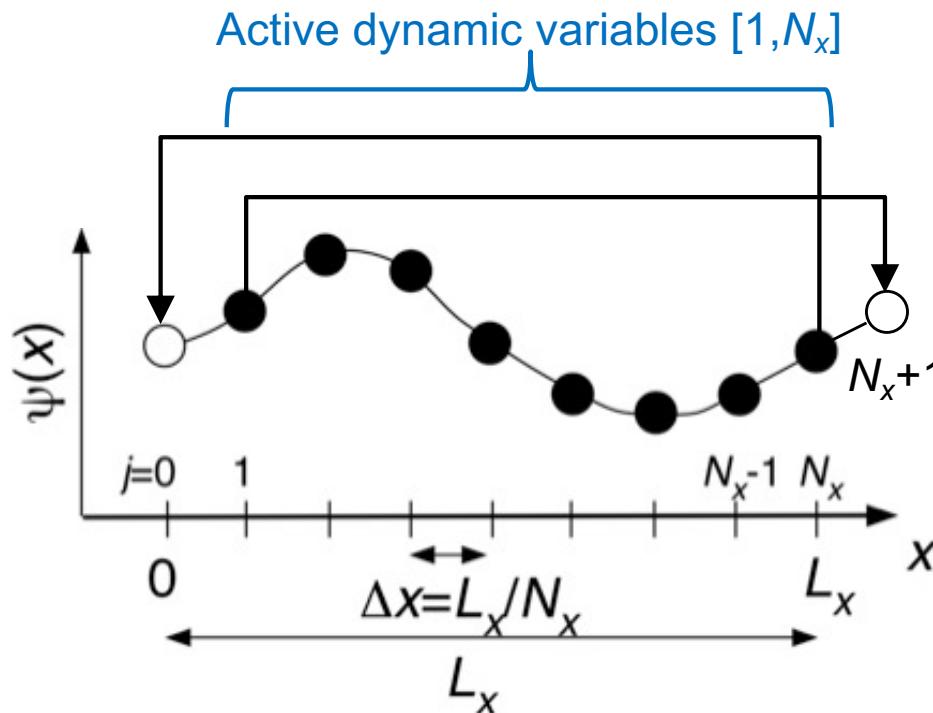
$$D = \begin{bmatrix} \varepsilon_+ & 0 \\ 0 & \varepsilon_- \end{bmatrix}$$

$$D^n = \begin{bmatrix} \varepsilon_+^n & 0 \\ 0 & \varepsilon_-^n \end{bmatrix}$$

Data Structures in Program qd1.c

- Wave function: $\text{psi}[NX+2][2]$ $\text{psi}[j][0|1] = (\text{Re}|\text{Im})\psi_{j\Delta x}$
- Periodic boundary condition by auxiliary elements

```
for (s=0; s<=1; s++) {  
    psi[0][s] = psi[NX][s];  
    psi[NX+1][s] = psi[1][s];  
}
```



Potential Propagator in qd1.c

- Potential barrier: $v[NX+2]$
- Potential propagator: $\exp(-iV\Delta t/2)$, $u[NX+2][2]$
- Potential propagation: $\psi \leftarrow \exp(-iV\Delta t/2)\psi$

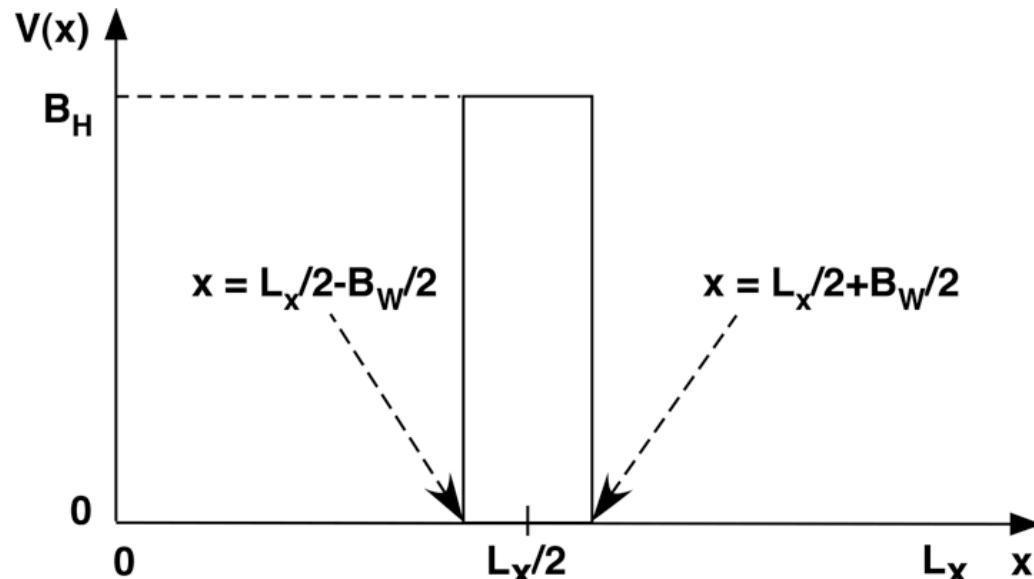
```

for (sx=1; sx<=NX; sx++)
    wr=u[sx][0]*psi[sx][0]-u[sx][1]*psi[sx][1];
    wi=u[sx][0]*psi[sx][1]+u[sx][1]*psi[sx][0];
    psi[sx][0]=wr;
    psi[sx][1]=wi;
}

```

$$\begin{cases} u[j][0] = \cos\left(-\frac{\Delta}{2}V_j\right) \\ u[j][1] = \sin\left(-\frac{\Delta}{2}V_j\right) \end{cases}$$

$$\exp\left(-\frac{iV_j\Delta}{2}\right)\psi_j \equiv u\psi = (u_0 + iu_1)(\psi_0 + i\psi_1) = \overbrace{(u_0\psi_0 - u_1\psi_1)}^{\text{new } \psi_0} + i\overbrace{(u_0\psi_1 + u_1\psi_0)}^{\text{new } \psi_1}$$



Kinetic Propagator in qd1.c

$$\begin{aligned} \left(U_x^{(\text{half})} \psi \right)_i &= \varepsilon_2^- \delta_{\text{mod}(i,2),0} \psi_{i-1} + \varepsilon_2^+ \psi_i + \varepsilon_2^- \delta_{\text{mod}(i,2),1} \psi_{i+1} \\ \left(U_x^{(\text{full})} \psi \right)_i &= \varepsilon_1^- \delta_{\text{mod}(i,2),1} \psi_{i-1} + \varepsilon_1^+ \psi_i + \varepsilon_1^- \delta_{\text{mod}(i,2),0} \psi_{i+1} \end{aligned}$$

```

for (sx=1; sx<=NX; sx++) { // wrk[][]|psi[][] holds new|old wave function
    wr=al[t][0]*psi[sx][0]-al[t][1]*psi[sx][1]; // al[0|1][][]: αhalf|full
    wi=al[t][0]*psi[sx][1]+al[t][1]*psi[sx][0];
    wr+=(bl[t][sx][0]*psi[sx-1][0]-bl[t][sx][1]*psi[sx-1][1]); // bl[0|1][][]: βhalf|full(low)
    wi+=(bl[t][sx][0]*psi[sx-1][1]+bl[t][sx][1]*psi[sx-1][0]);
    wr+=(bu[t][sx][0]*psi[sx+1][0]-bu[t][sx][1]*psi[sx+1][1]); // bu[0|1][][]: βhalf|full(high)
    wi+=(bu[t][sx][0]*psi[sx+1][1]+bu[t][sx][1]*psi[sx+1][0]);
    wrk[sx][0]=wr;
    wrk[sx][1]=wi; }  $\psi_j \leftarrow \beta_l \psi_{j-1} + \alpha \psi_j + \beta_u \psi_{j+1}$ 

```

for (sx=1; sx<=NX; sx++) // Copy new wave function back to psi

```

for (s=0; s<=1; s++)
    psi[sx][s]=wrk[sx][s];

```

$$\exp(-i\Delta t T_x) = U_x^{(\text{half})} U_x^{(\text{full})} U_x^{(\text{half})} + O([\Delta t]^3) \quad \begin{cases} \varepsilon_n^+ = \frac{1}{2} \left[\exp\left(-\frac{i\Delta t}{n}(a+b)\right) + \exp\left(-\frac{i\Delta t}{n}(a-b)\right) \right] \\ \varepsilon_n^- = \frac{1}{2} \left[\exp\left(-\frac{i\Delta t}{n}(a+b)\right) - \exp\left(-\frac{i\Delta t}{n}(a-b)\right) \right] \end{cases}$$

$$= \begin{bmatrix} \varepsilon_2^+ & \varepsilon_2^- \\ \varepsilon_2^- & \varepsilon_2^+ \\ & \ddots \\ & & \ddots \\ & & & \varepsilon_2^+ & \varepsilon_2^- \\ & & & \varepsilon_2^- & \varepsilon_2^+ \\ & & & \ddots & \\ & & & & \varepsilon_2^+ & \varepsilon_2^- \\ & & & & \varepsilon_2^- & \varepsilon_2^+ \\ & & & & \ddots & \\ & & & & & \varepsilon_2^+ & \varepsilon_2^- \\ & & & & & \varepsilon_2^- & \varepsilon_2^+ \end{bmatrix} \begin{bmatrix} \varepsilon_1^+ & & & & & \\ & \varepsilon_1^+ & \varepsilon_1^- & & & \\ & \varepsilon_1^- & \varepsilon_1^+ & \ddots & & \\ & & & \varepsilon_1^+ & \varepsilon_1^- & \\ & & & \varepsilon_1^- & \varepsilon_1^+ & \\ & & & & & \varepsilon_1^+ \end{bmatrix} \begin{bmatrix} \varepsilon_2^+ & \varepsilon_2^- \\ \varepsilon_2^- & \varepsilon_2^+ \\ & \ddots \\ & & \ddots \\ & & & \varepsilon_2^+ & \varepsilon_2^- \\ & & & \varepsilon_2^- & \varepsilon_2^+ \\ & & & \ddots & \\ & & & & \varepsilon_2^+ & \varepsilon_2^- \\ & & & & \varepsilon_2^- & \varepsilon_2^+ \\ & & & & \ddots & \\ & & & & & \varepsilon_2^+ & \varepsilon_2^- \\ & & & & & \varepsilon_2^- & \varepsilon_2^+ \end{bmatrix}$$

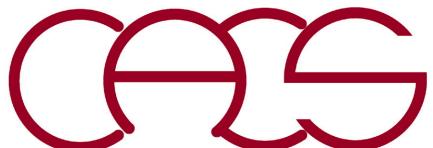
Quantum Dynamics—II

Spectral Method

Aiichiro Nakano

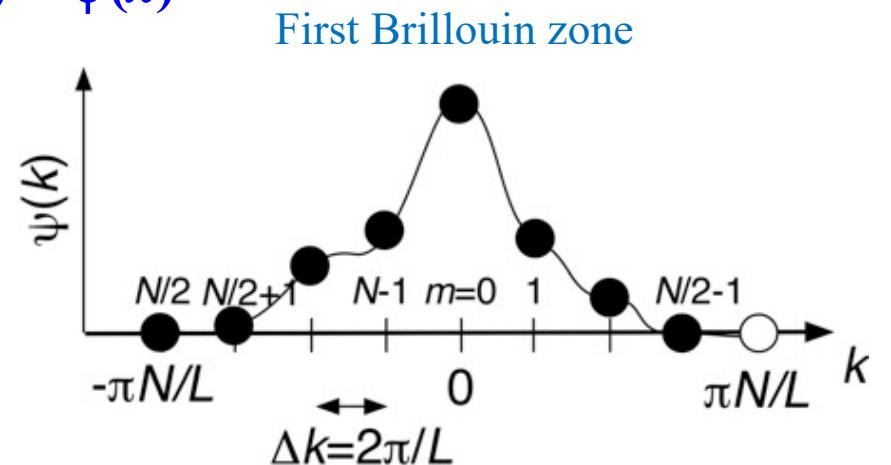
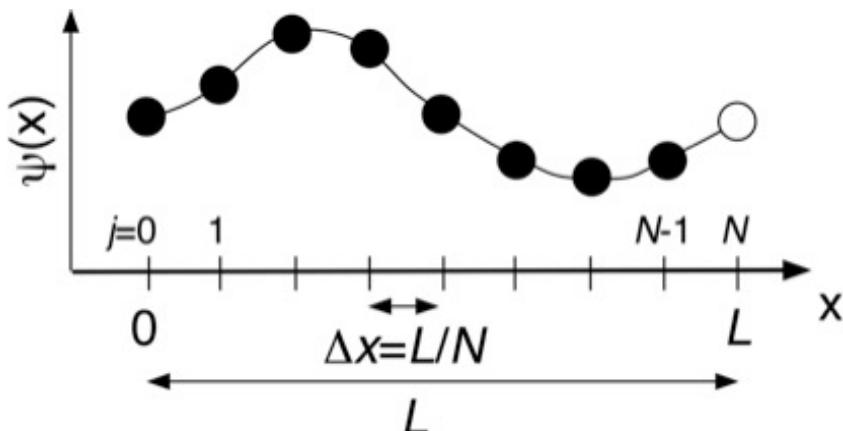
*Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Quantitative & Computational Biology
University of Southern California*

Email: anakano@usc.edu



Discrete Fourier Transform

- Discretize $\psi(x) \in C (x \in [0, L])$ on N mesh points, $x_j = j\Delta x (j = 0, \dots, N-1)$, with equal mesh spacing, $\Delta x = L/N$
- Periodic boundary condition: $\psi(x + L) = \psi(x)$

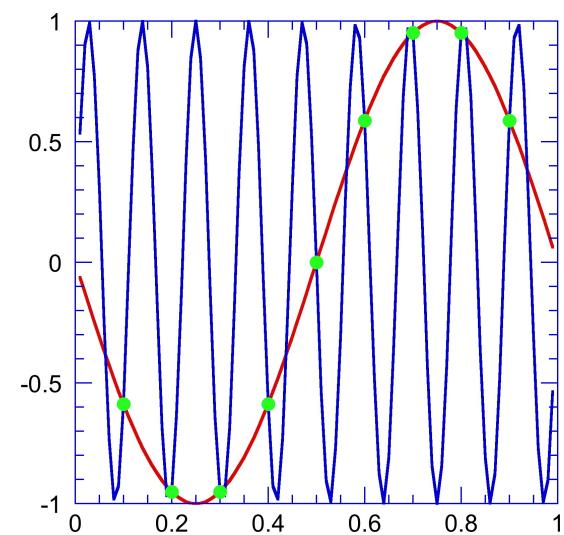


- Discrete Fourier transform: Represents $\psi(x)$ as a linear combination of $\exp(ikx) = \cos(kx) + i \sin(kx)$, with different wave numbers, k

$$\psi_j = \sum_{m=0}^{N-1} \tilde{\psi}_m \exp(i k_m x_j)$$

$$k_m = \begin{cases} 2\pi m / L & (m = 0, 1, \dots, N/2 - 1) \\ 2\pi(m - N)/L & (m = N/2, N/2 + 1, \dots, N - 1) \end{cases}$$

$$\tilde{\psi}_m = \frac{1}{N} \sum_{j=0}^{N-1} \psi_j \exp(-i k_m x_j)$$



Wave Numbers

- Periodic boundary condition, $\psi(x + L) = \psi(x)$, is guaranteed by choosing $k_m = 2\pi m/L$

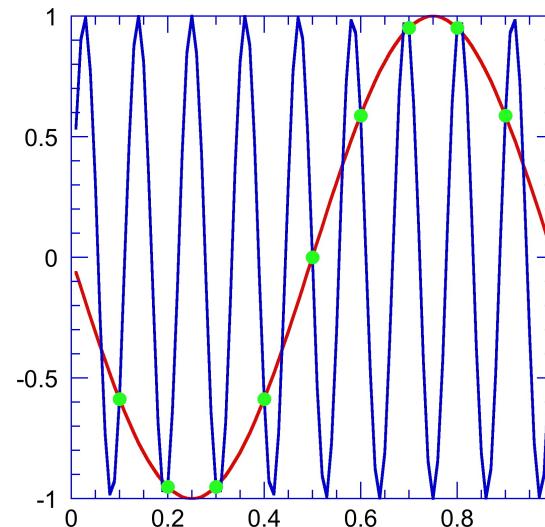
$$e^{i\left(\frac{2\pi m}{L}(x+L)\right)} = e^{i\left(\frac{2\pi m}{L}x + 2\pi m\right)} = e^{i\frac{2\pi m}{L}x} \underbrace{e^{i2\pi m}}_{(e^{i2\pi})^m=1^m=1} = e^{i\frac{2\pi m}{L}x}$$

- Folding back the latter half of wave numbers by $\frac{2\pi N}{L} = 2\pi/\Delta x$ (cf. first Brillouin zone):

From $[0, \frac{2\pi N}{L}] = [0, \frac{2\pi}{\Delta x}]$ to $\left[-\frac{2\pi}{L} \cdot \frac{N}{2}, \frac{2\pi}{L} \cdot \frac{N}{2}\right] = \left[-\frac{\pi}{\Delta x}, \frac{\pi}{\Delta x}\right]$

The shift won't change wave-function value on any grid point

$$\begin{aligned} & e^{i\left(k_m - \frac{2\pi N}{L}\right)x_j} \\ &= e^{ik_m x_j} e^{-i\frac{2\pi N}{L} \cdot \frac{L}{N} j} \\ &= e^{ik_m x_j} e^{\overbrace{-i2\pi j}^1} \\ &= e^{i k_m x_j} \end{aligned}$$



Orthonormal Basis Set

- **N -dimensional vector space:** $|\psi\rangle = (\psi_0, \psi_1, \dots, \psi_{N-1})$
- **Plane-wave basis set:** $\left\{ |m\rangle = b_m(x_j) = \frac{1}{\sqrt{N}} \exp(i k_m x_j) \mid m = 0, 1, \dots, N-1 \right\}$
- **Orthonormality:** $\langle m|n \rangle = \sum_{j=0}^{N-1} b_m^*(x_j) b_n(x_j) = \delta_{m,n} = \begin{cases} 1 & (m = n) \\ 0 & (m \neq n) \end{cases}$

$$\therefore \langle m|n \rangle = \frac{1}{N} \sum_{j=0}^{N-1} \exp(i(k_n - k_m)x_j) = \frac{1}{N} \sum_{j=0}^{N-1} \exp\left(i \frac{2\pi}{N} (n-m)j\right)$$

$$(k_n - k_m)x_j = \frac{2\pi(n-m)}{L} \cdot \frac{L}{N} j$$

$$= \begin{cases} \frac{1}{N} \frac{\exp(i2\pi(n-m))-1}{\exp(i\frac{2\pi}{N}(n-m))-1} = 0 & (m \neq n) \\ \frac{1}{N} \cdot N = 1 & (m = n) \end{cases}$$

Geometric series

$$S = \sum_{j=0}^{N-1} \left(e^{i\frac{2\pi}{N}(n-m)} \right)^j = 1 + \dots + r^{N-1}$$

$$rS = r + \dots + r^N$$

$$\therefore (r-1)S = (r^N - 1)$$

$$|\psi\rangle = \sum_m c_m |m\rangle$$

$$\langle n| \times \Downarrow$$

- **Completeness:** $|\psi\rangle = \sum_{m=0}^{N-1} |m\rangle \langle m|\psi\rangle$ or $1 = \sum_{m=0}^{N-1} |m\rangle \langle m|$
- **Fourier transform:** $\psi_j = \sum_{m=0}^{N-1} \exp(i k_m x_j) \boxed{\frac{1}{N} \sum_{l=0}^{N-1} \exp(-i k_m x_l) \psi_l}$

$\tilde{\psi}_m$

Spectral Method

- Kinetic-energy operator is diagonal in the momentum space: $\tilde{\psi}_m \xrightarrow{T} \frac{k_m^2}{2} \tilde{\psi}_m$

$$-\frac{1}{2} \frac{\partial^2}{\partial x^2} \sum_{m=0}^{N-1} \tilde{\psi}_m \exp(ik_m x) = \sum_{m=0}^{N-1} \frac{k_m^2}{2} \tilde{\psi}_m \exp(ik_m x)$$

- Potential-energy operator is diagonal in the real space: $\psi_j \xrightarrow{V} V_j \psi_j$
- Split-operator technique & spectral method

$$\psi(t + \Delta t) = \exp\left(-\frac{iV\Delta t}{2}\right) \xrightarrow{F} \exp(-iT\Delta t) \xrightarrow{F^{-1}} \exp\left(-\frac{iV\Delta t}{2}\right) \psi(t) + O((\Delta t)^3)$$

1. $\psi_j \xrightarrow{\exp(-iV\Delta t/2)} \exp(-iV_j \Delta t / 2) \psi_j \quad O(N)$

2. $\psi_j \xrightarrow{F^{-1}} F^{-1} \psi_j = \tilde{\psi}_m = \frac{1}{N} \sum_{j=1}^N \psi_j \exp(-ik_m x_j) \quad O(N^2)?$

3. $\tilde{\psi}_m \xrightarrow{\exp(-iT\Delta t)} \exp(-ik_m^2 \Delta t / 2) \tilde{\psi}_m \quad O(N)$

4. $\tilde{\psi}_m \xrightarrow{F} F \tilde{\psi}_m = \psi_j = \sum_{m=1}^N \tilde{\psi}_m \exp(ik_m x_j) \quad O(N^2)?$

5. $\psi_j \xrightarrow{\exp(-iV\Delta t/2)} \exp(-iV_j \Delta t / 2) \psi_j \quad O(N)$

Exact exponentiation!

$$\begin{aligned} & \exp\left(\frac{i\Delta t}{2} \frac{\partial^2}{\partial x^2}\right) \sum_m \tilde{\psi}_m e^{-ik_m x} \\ &= \sum_m \exp\left(-\frac{i\Delta t k_m^2}{2}\right) \tilde{\psi}_m e^{-ik_m x} \end{aligned}$$

Solution: Fast Fourier Transform

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):

- 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 *IEEE Comput. Sci. Eng.* **2**(1), 22 ('00)
- Fast Multipole Method

Fast Fourier Transform

- Danielson-Lanczos algorithm:

$$\begin{aligned}\psi_j &= \sum_{m=0}^{N-1} \tilde{\psi}_m \exp(ik_m x_j) = \sum_{m=0}^{N-1} \tilde{\psi}_m \exp(i2\pi m j / N) \quad O(N^2)! \\ &= \sum_{m=0}^{N/2-1} \tilde{\psi}_{2m} \exp(i2\pi(2m)j / N) + \sum_{m=0}^{N/2-1} \tilde{\psi}_{2m+1} \exp(i2\pi(2m+1)j / N) \\ &\quad \text{even terms} \qquad \qquad \qquad \text{odd terms} \\ &= \sum_{m=0}^{N/2-1} \tilde{\psi}_{2m} \exp(i2\pi m j / (N/2)) + \exp(i2\pi j / N) \sum_{m=0}^{N/2-1} \tilde{\psi}_{2m+1} \exp(i2\pi m j / (N/2))\end{aligned}$$

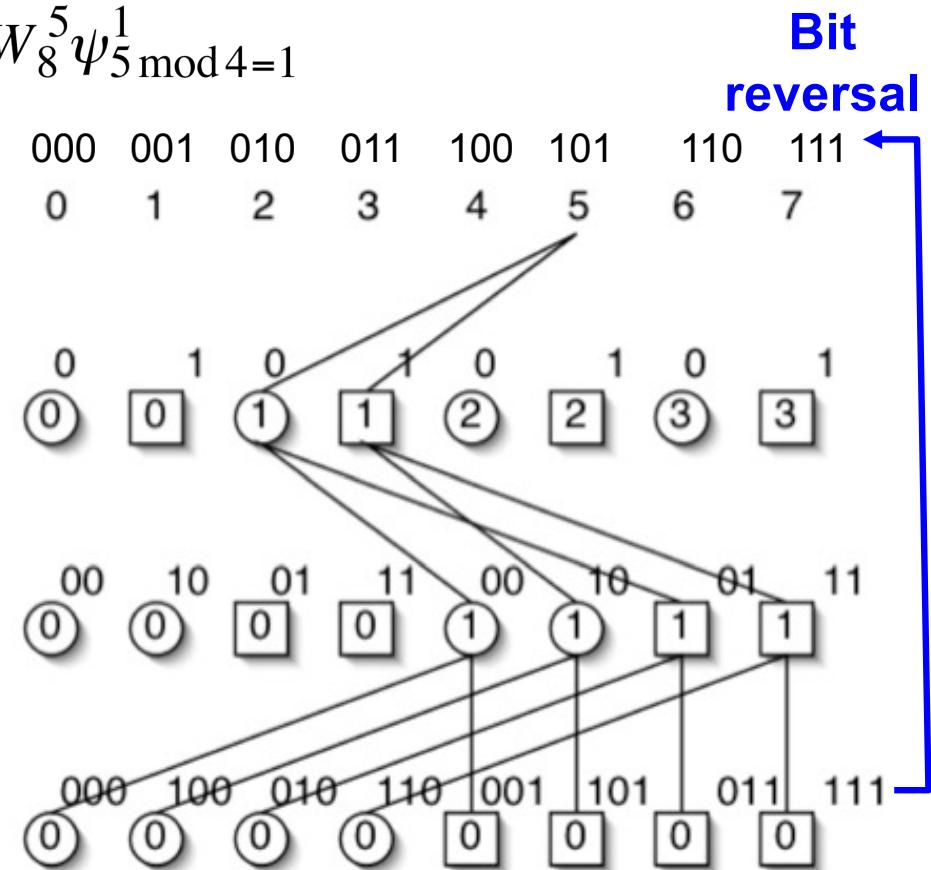
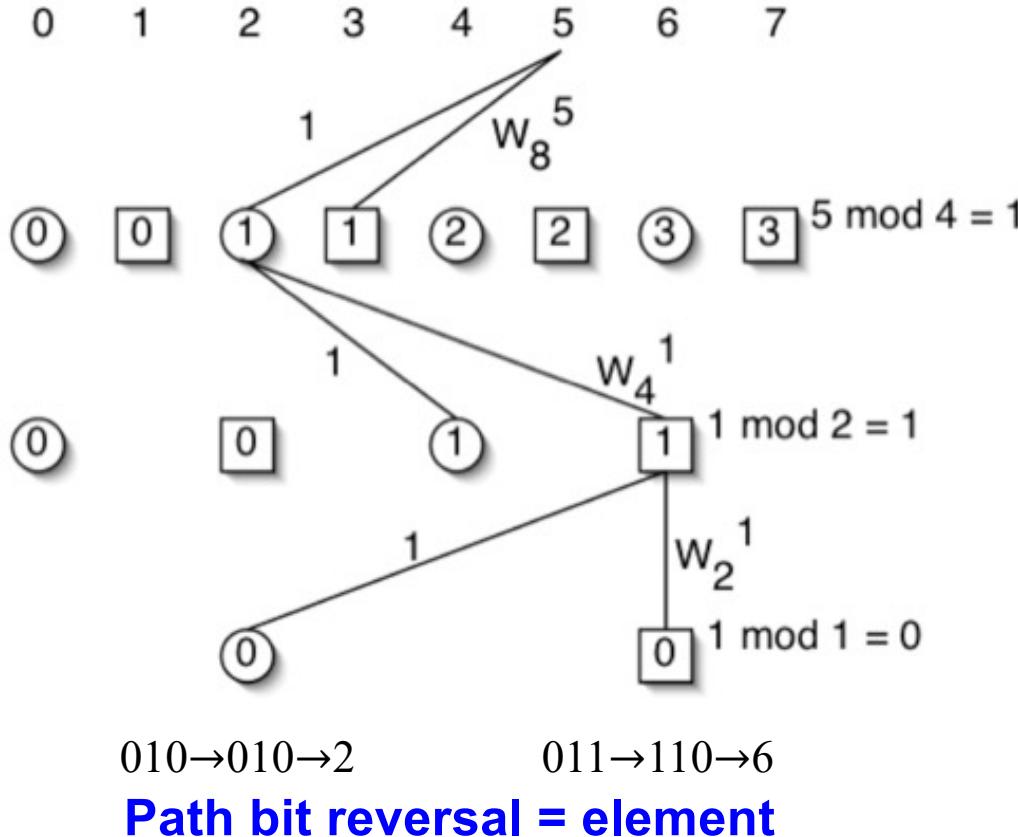
$$\left. \begin{array}{l} \psi_j^0 = \sum_{m=0}^{N/2-1} \tilde{\psi}_{2m} \exp(i2\pi m j / (N/2)) \\ \psi_j^1 = \sum_{m=0}^{N/2-1} \tilde{\psi}_{2m+1} \exp(i2\pi m j / (N/2)) \\ W_N = \exp(i2\pi / N) \end{array} \right\} \begin{array}{l} \text{even & odd} \\ \text{subarray} \\ \text{Fourier} \\ \text{decompositions} \\ j \text{ read as } j \bmod N/2 \end{array}$$

Divide-and-conquer

Fast Fourier Transform

- Recursive sub-Fourier transforms: $\psi_j = \psi_j^0 + W_N^j \psi_j^1$

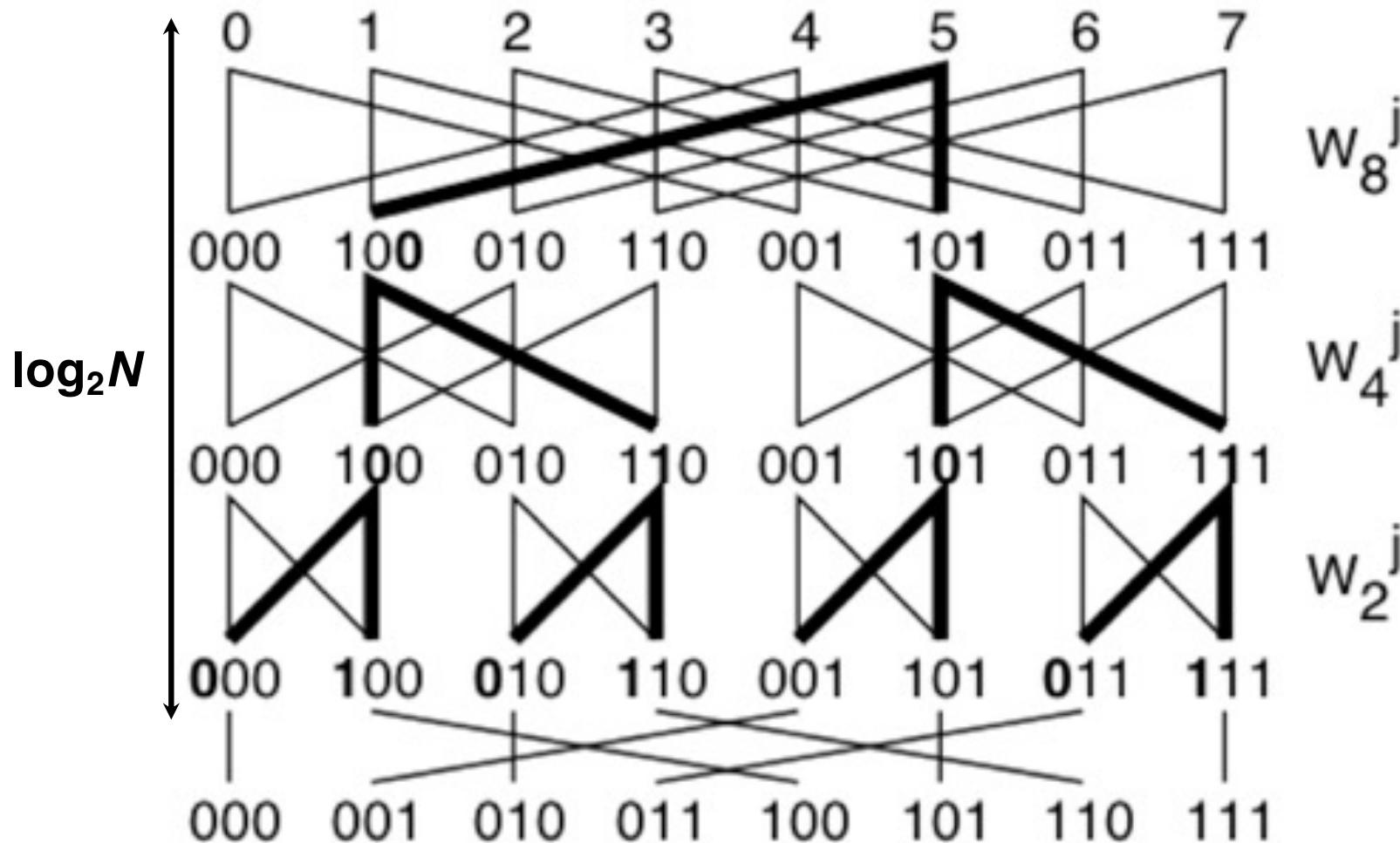
$$\psi_5 = \psi_{5 \bmod 4=1}^0 + W_8^5 \psi_{5 \bmod 4=1}^1$$



Recursion tree
 $O(N)$ operations per element

Fast Fourier Transform Algorithm

- Many computations are shared among the recursion trees
- Butterfly (hypercube) data exchange after bit-reversal
- $2N\log_2 N$ complex arithmetic operations

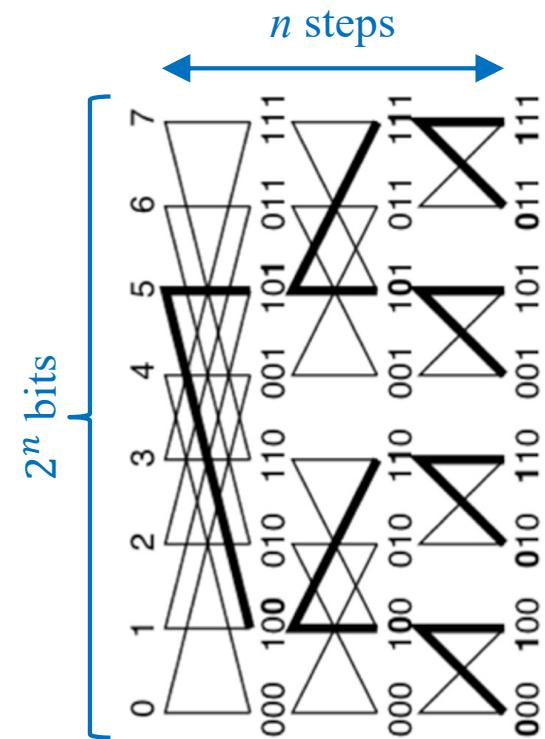
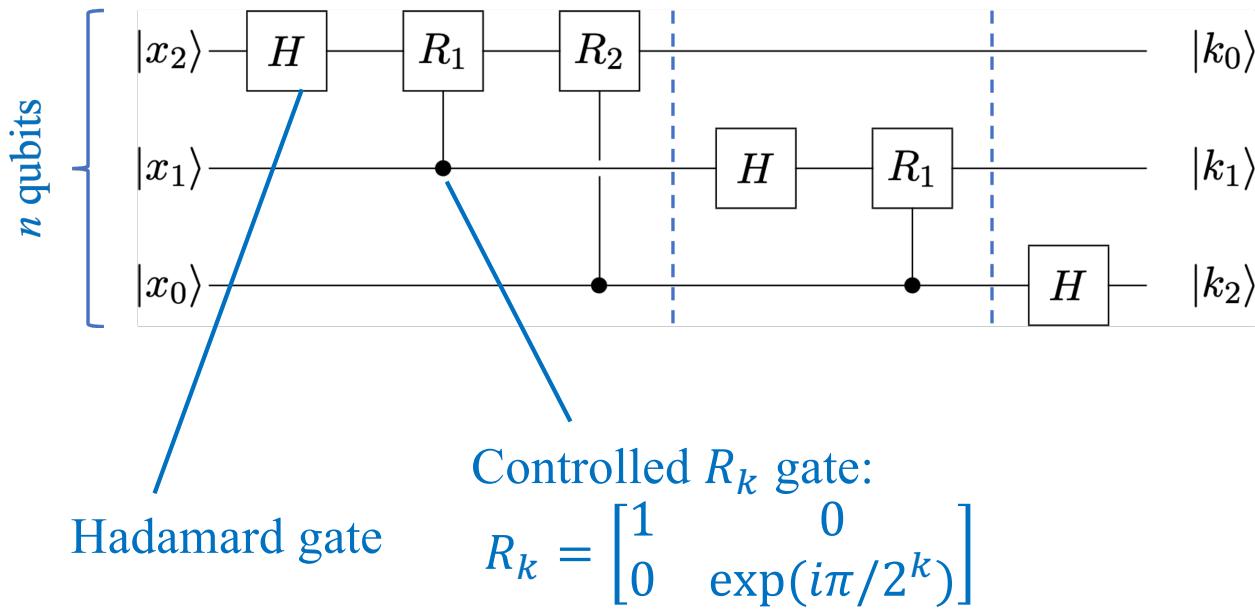


Quantum Fourier Transform (QFT)

- On a quantum computer, quantum parallelism allows Fourier transform to be performed using only n qubits & $O(n^2)$ gates for $N = 2^n$

$$\text{QFT: } |x\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{i \frac{2\pi k x}{N}} |k\rangle$$

$$n + (n - 1) + \dots + 2 + 1 = \frac{n(n+1)}{2} \text{ gates (depths)}$$



- Compare QFT with $2N\log_2 N = O(2^n n)$ arithmetic operations in classical fast Fourier transform (FFT):

Exponential operation reduction: $O(2^n n / n^2) = O(2^n / n)$

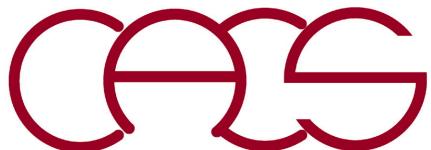
Parallel Quantum Dynamics

Aiichiro Nakano

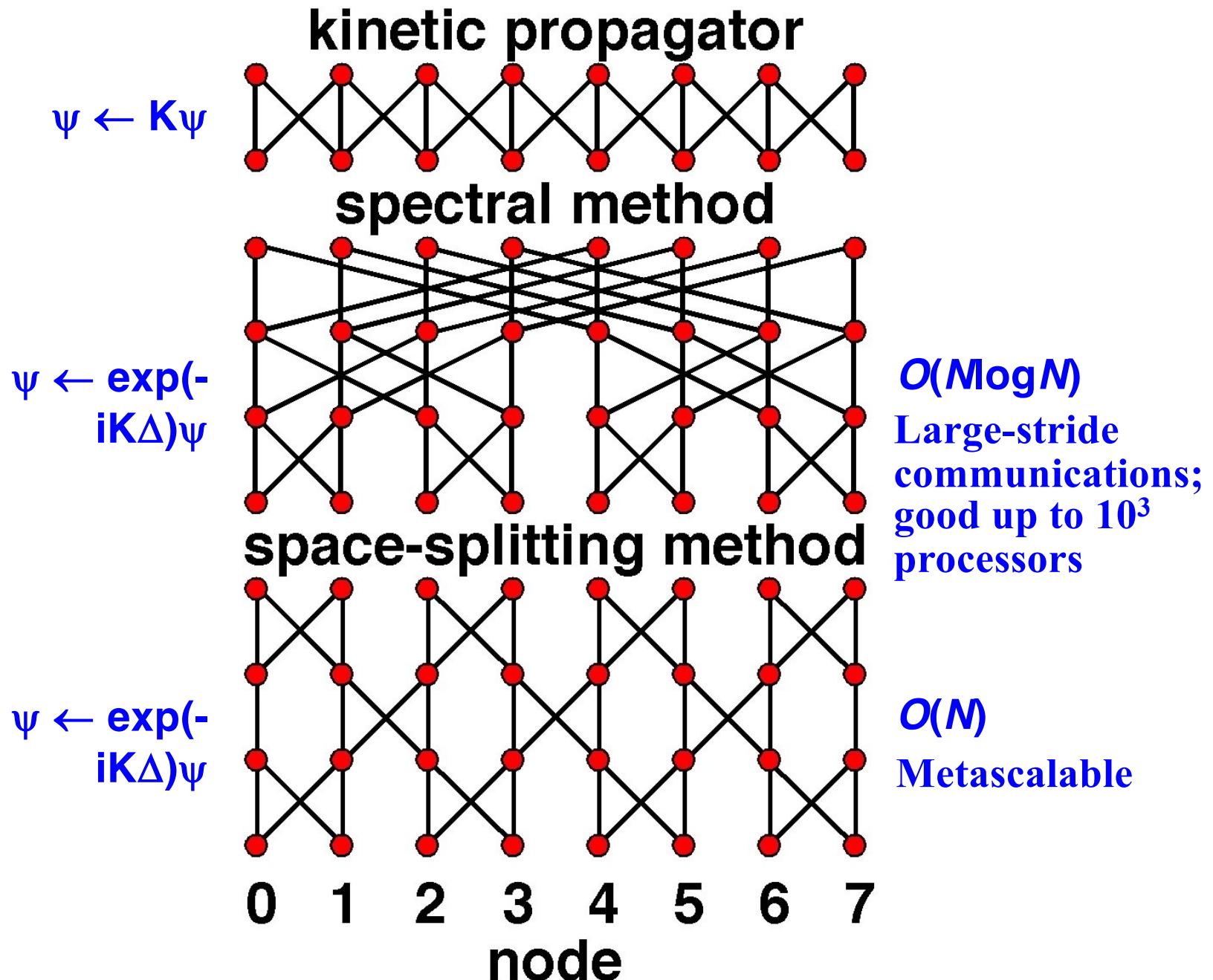
*Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Quantitative & Computational Biology
University of Southern California*

Email: anakano@usc.edu

Self-centric parallelization of a partial-differential-equation solver
as a ‘boundary condition’



Parallel QD Communications



Parallel QD Algorithms

- Not all algorithms are scalable on parallel computers
- Implicit solvers (e.g. Crank-Nicholson method) are numerically stable but less scalable due to sequential dependence

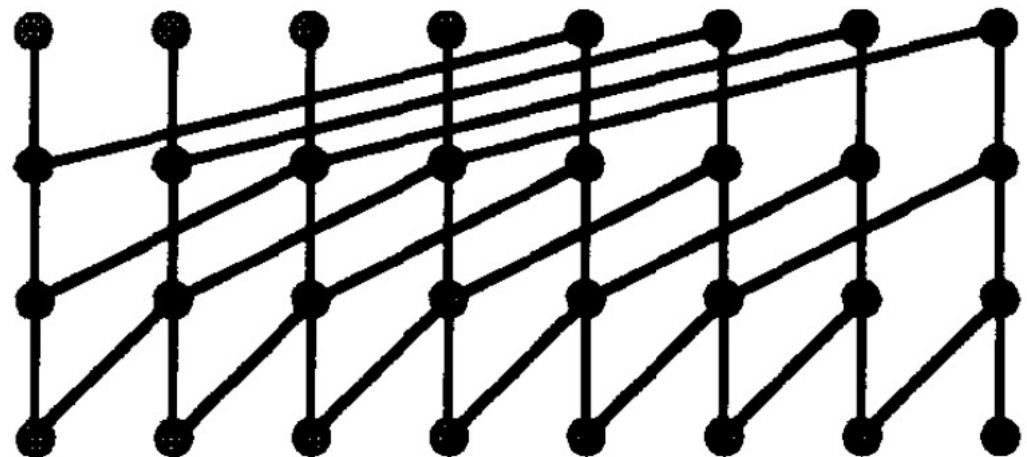
$$\psi(t + \Delta t) \leftarrow \exp\left(-\frac{i}{\hbar} \hat{H} \Delta t\right) \psi(t) \cong \frac{1 - \frac{i}{2\hbar} \hat{H} \Delta t}{1 + \frac{i}{2\hbar} \hat{H} \Delta t} \psi(t) + O((\Delta t)^3)$$

$$\overbrace{\left(1 + \frac{i}{2\hbar} \hat{H} \Delta t\right)}^A \overbrace{\psi(t + \Delta t)}^x = \overbrace{\left(1 - \frac{i}{2\hbar} \hat{H} \Delta t\right)}^b \psi(t)$$

$$\alpha x_{i-1} + \beta x_i + \alpha x_{i+1} = b_i$$

⇒

$$x_{i+1} \leftarrow \frac{1}{\alpha} b_i - \frac{\beta}{\alpha} x_i - x_{i-1}$$



- Sequential recursion needs be converted to divide-&-conquer (recursive doubling) for parallelization

Self-Centric (SC) Parallelization

- SC is the easiest serial-to-parallel migration path *via* single-program multiple-data (SPMD) programming
 1. Take a serial code
 2. Each MPI rank only works on a spatial subsystem
 3. Boundary information obtained from neighbor ranks
 4. Long-range information (if needed) by divide-&-conquer, like real-space multigrids; scalability behavior similar to short-ranged

F. Shimojo *et al.*, *J. Chem. Phys.* 140, 18A529 ('14)
K. Nomura *et al.*, *IEEE/ACM Supercomputing, SC14* ('14)
A. Nakano, *Comput. Phys. Commun.* 104, 59 ('97)



Quantum Dynamics Program:qd1.c

```
for step = 1 to NSTEP
```

```
pot_prop():  $\psi_j \leftarrow \exp(-iV_j \Delta t / 2) \psi_j$  ( $j \in [1, NX]$ )
```

```
kin_prop( $\Delta t / 2$ )
```

```
kin_prop( $\Delta t$ )
```

```
kin_prop( $\Delta t / 2$ )
```

```
pot_prop():  $\psi_j \leftarrow \exp(-iV_j \Delta t / 2) \psi_j$  ( $j \in [1, NX]$ )
```

$$\begin{aligned}\psi(t + \Delta t) &\leftarrow \exp(-iV\Delta t / 2) \exp(-iT_x\Delta t) \exp(-iV\Delta t / 2) \psi(t) \\ &= e^{-iV\Delta t / 2} U_x^{(\text{half})} U_x^{(\text{full})} U_x^{(\text{half})} e^{-iV\Delta t / 2} \psi(t)\end{aligned}$$

```
kin_prop( $\Delta$ )
```

```
periodic_bc():  $\psi_0 \leftarrow \psi_{NX}; \psi_{NX+1} \leftarrow \psi_1$ 
```

```
for  $\forall j \in [1, NX]$ 
```

```
 $\psi_j \leftarrow blx(\Delta)_j \psi_{j-1} + al(\Delta)_j \psi_j + bux(\Delta)_j \psi_{j+1}$ 
```

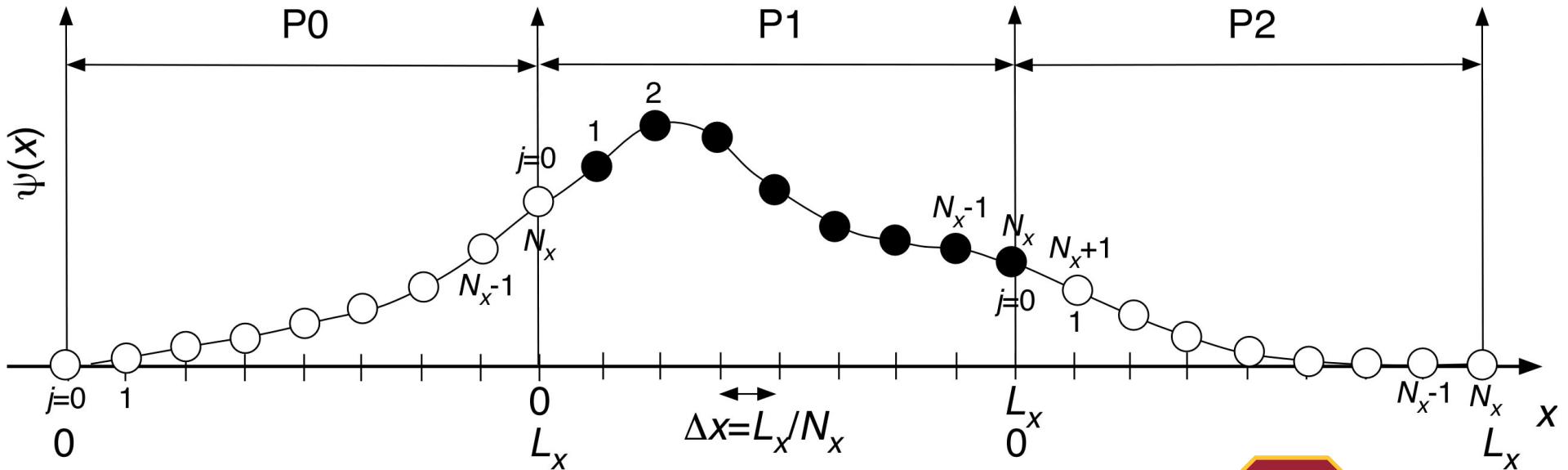
$$\begin{cases} \varepsilon_n^+ = \frac{1}{2} \left[\exp\left(-\frac{i\Delta t}{n}(a+b)\right) + \exp\left(-\frac{i\Delta t}{n}(a-b)\right) \right] \\ \varepsilon_n^- = \frac{1}{2} \left[\exp\left(-\frac{i\Delta t}{n}(a+b)\right) - \exp\left(-\frac{i\Delta t}{n}(a-b)\right) \right] \end{cases}$$

$$\exp(-i\Delta t T_x) \cong U_x^{(\text{half})} U_x^{(\text{full})} U_x^{(\text{half})} = \begin{bmatrix} \varepsilon_2^+ & \varepsilon_2^- \\ \varepsilon_2^- & \varepsilon_2^+ \\ & \ddots & \ddots & \ddots \\ & & \varepsilon_2^+ & \varepsilon_2^- \\ & & \varepsilon_2^- & \varepsilon_2^+ \\ & & & \ddots & \ddots & \ddots \\ & & & & \varepsilon_1^+ & \varepsilon_1^- \\ & & & & \varepsilon_1^- & \varepsilon_1^+ \\ & & & & & \ddots \\ & & & & & & \varepsilon_1^+ \\ & & & & & & & \ddots \\ & & & & & & & & \varepsilon_2^+ & \varepsilon_2^- \\ & & & & & & & & \varepsilon_2^- & \varepsilon_2^+ \\ & & & & & & & & & \ddots \\ & & & & & & & & & & \varepsilon_2^+ & \varepsilon_2^- \\ & & & & & & & & & & \varepsilon_2^- & \varepsilon_2^+ \end{bmatrix}$$

$\Psi_j(t+1) \leftarrow f(\Psi_{j-1}(t), \Psi_j(t), \Psi_{j+1}(t))$ ($j \in [1, NX]$)

SC Parallelization

- Self-centric spatial decomposition



- Local & global coordinates

$$\begin{cases} x_j = j\Delta x \\ x_j^{(\text{global})} = j\Delta x + pL_x \end{cases}$$

off-set



- Global coordinates only in `init_prop()` & `init_wavefn()`

$O(N)$ divide-&-conquer algorithms maximally expose data locality
→ easy self-centric parallelization & metascalable

Boundary Wave Function Caching

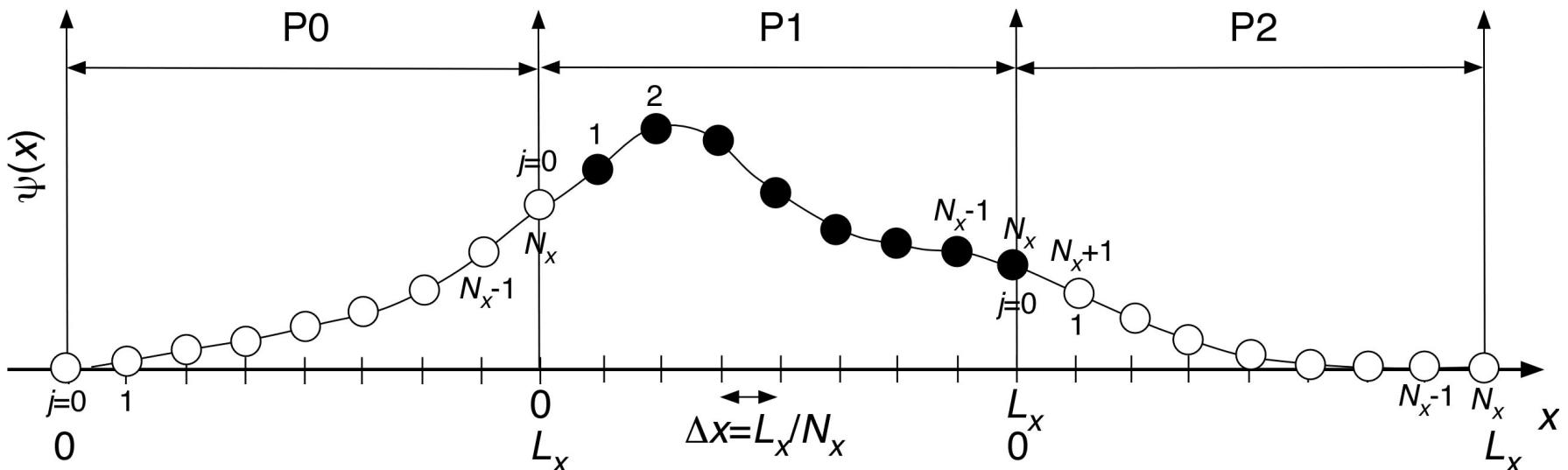
- Parallelized `periodic_bc()`

```
plw = (myid-1+nproc)%nproc; /* Lower partner process */
pup = (myid+1           )%nproc; /* Upper partner process */

/* Cache boundary wave function value at the lower end */
dbuf[0:1] ← psi[NX][0:1];
Send dbuf to pup;
Receive dbufr from plw;
psi[0][0:1] ← dbufr[0:1];
```

cf. serial_periodic_bc():
 $\Psi_0 \leftarrow \Psi_{NX}; \Psi_{NX+1} \leftarrow \Psi_1$

```
/* Cache boundary wave function value at the upper end */
dbuf[0:1] ← psi[1][0:1];
Send dbuf to plw;
Receive dbufr from pup;
psi[NX+1][0:1] ← dbufr[0:1];
```



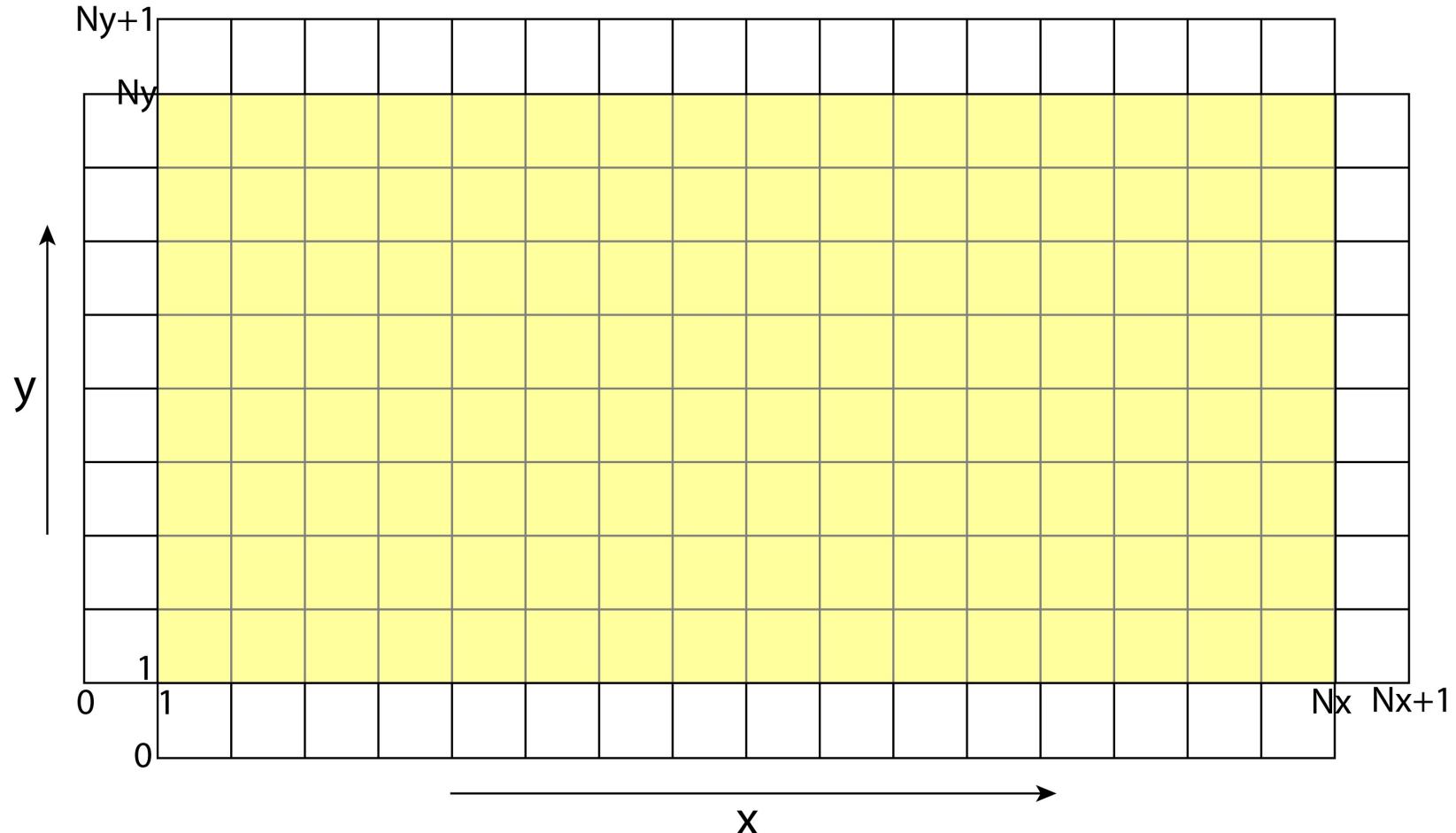
Multidimensional Parallelization

- Parallelized `periodic_bc()`

for \forall directions

send front row $\psi(\dots, 1 \text{ or } N_\alpha, \dots)$ to forward neighbor

receive back appendage $\psi(\dots, N_\alpha+1 \text{ or } 0, \dots)$ from back neighbor



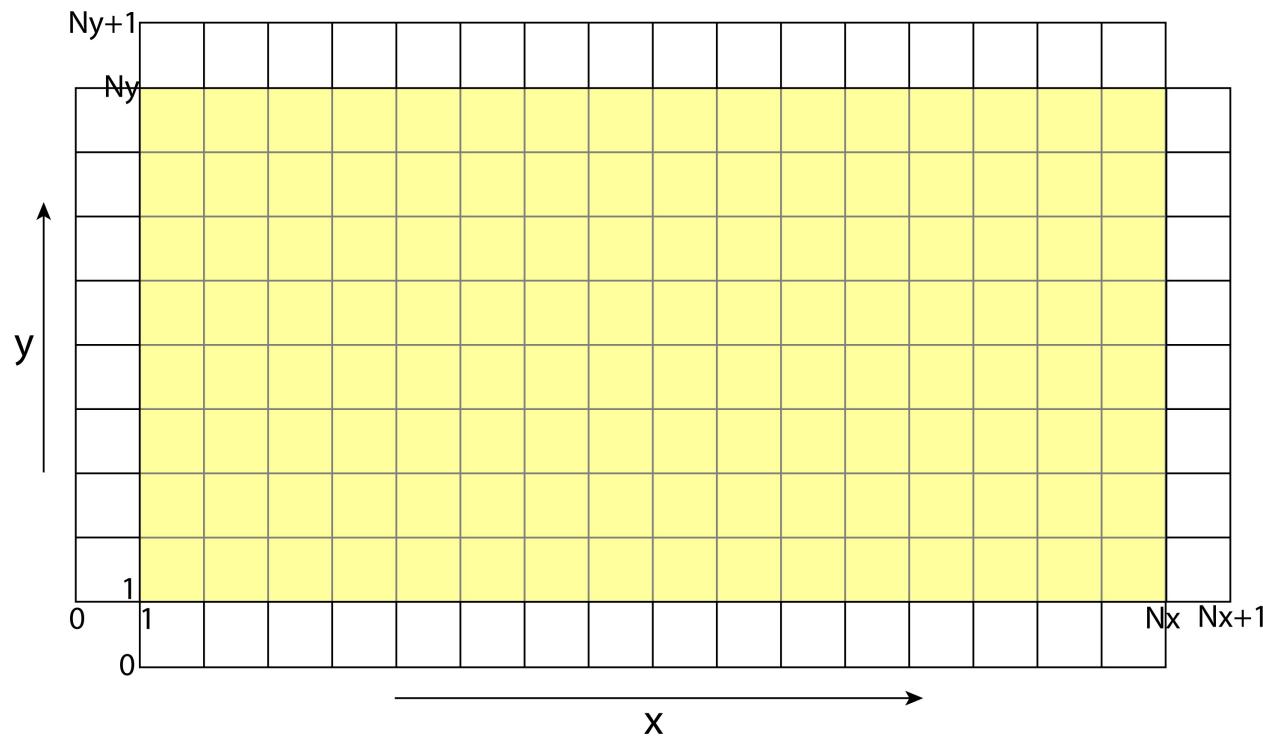
Multidimensional Parallelization

- Parameterized message composition

$$dbuf \leftarrow psi(i_b : i_e, j_b : j_e, k_b : k_e)$$
$$psi(i'_b : i'_e, j'_b : j'_e, k'_b : k'_e) \leftarrow dbufr$$

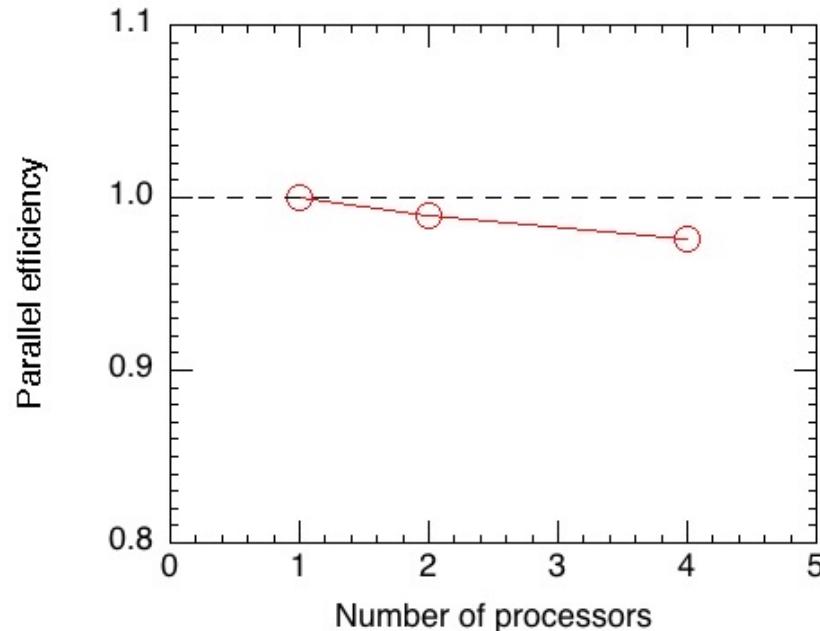
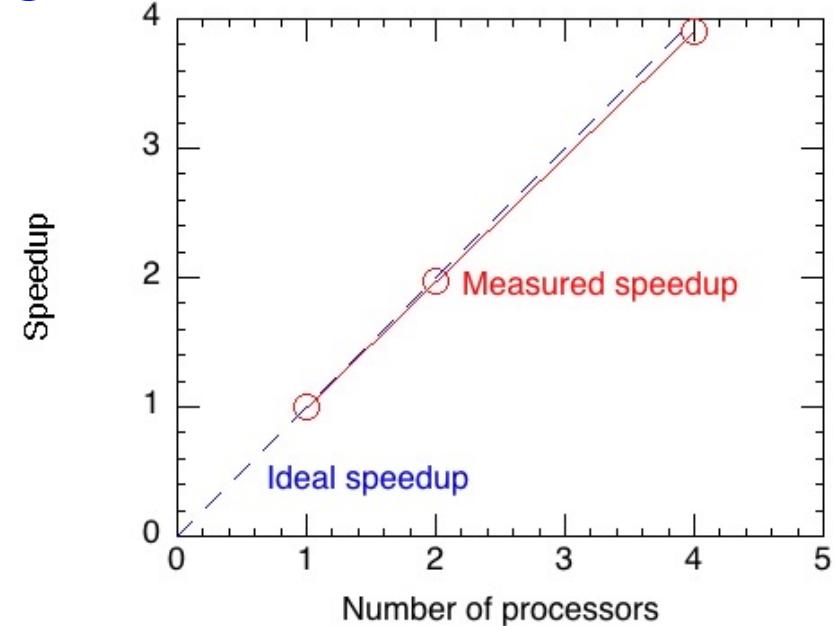
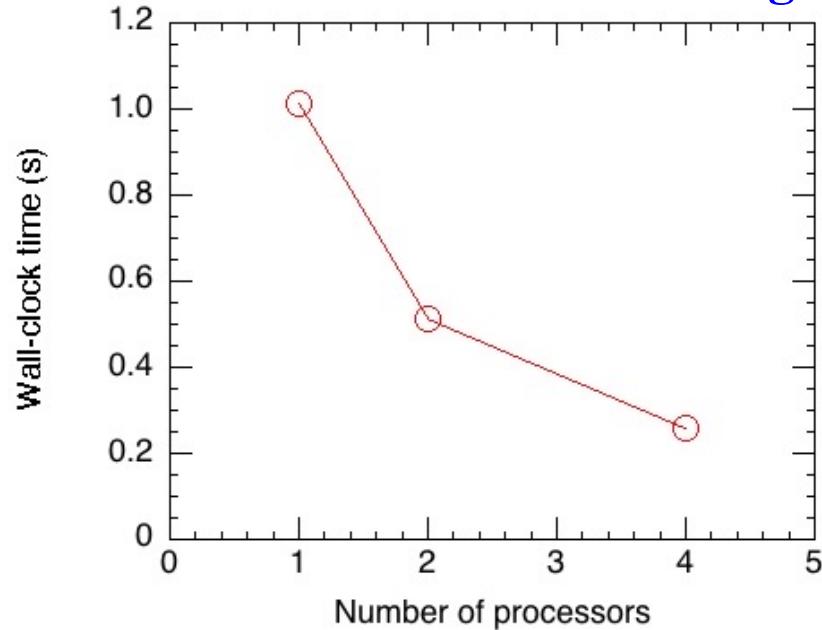
(Example) x-low direction

$$i_b = 1, i_e = 1, j_b = 1, j_e = N_y, k_b = 1, k_e = N_z$$
$$i'_b = N_x + 1, i'_e = N_x + 1, j'_b = 1, j'_e = N_y, k'_b = 1, k'_e = N_z$$



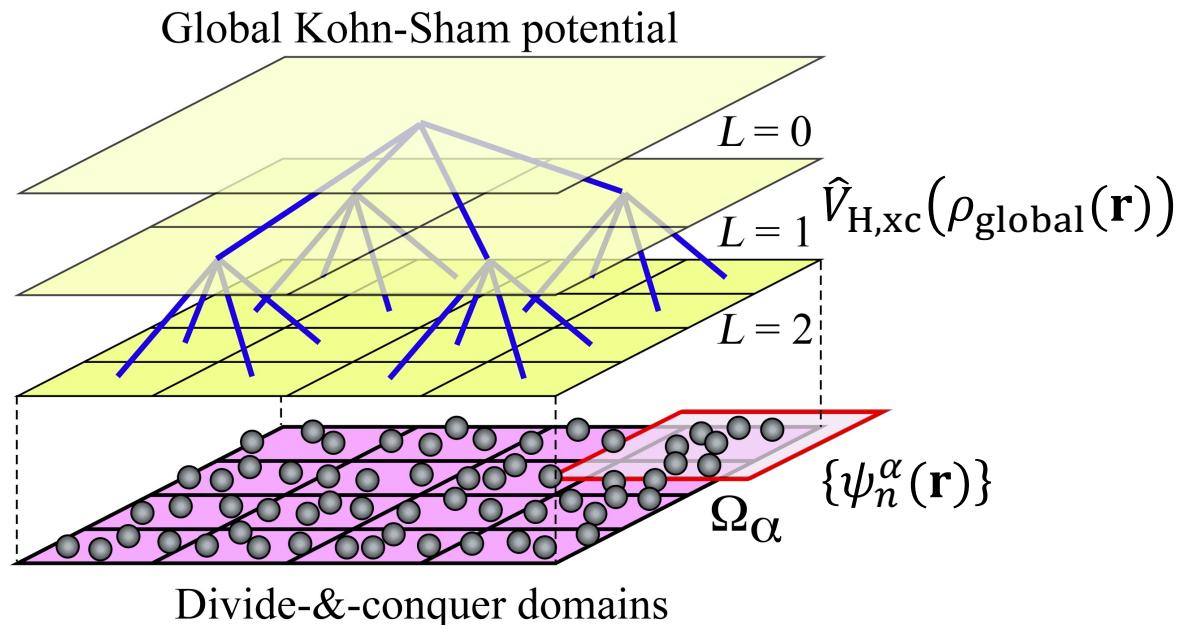
Parallel QD Results

Strong scaling results

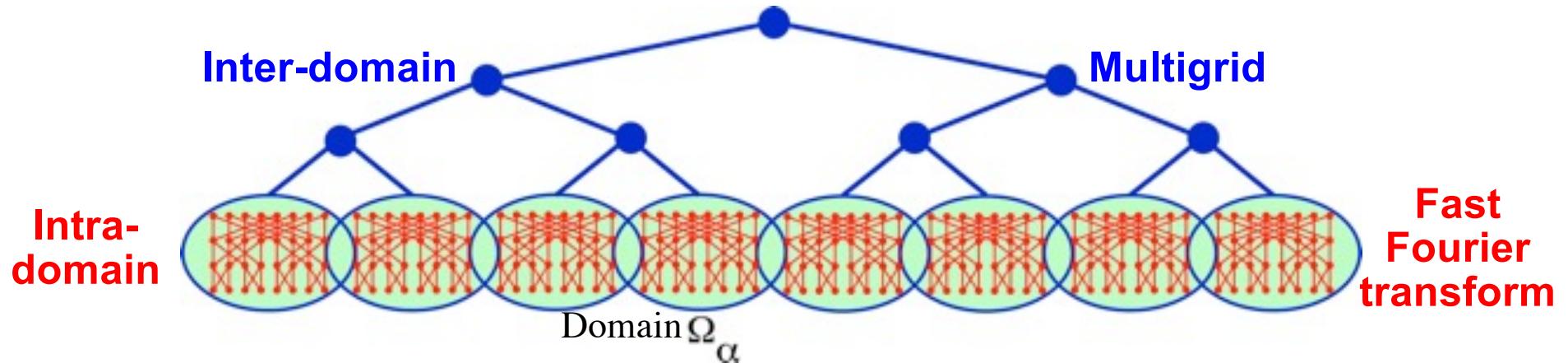


Hierarchical Parallel QD

- Divide-&-conquer density functional theory (DFT): Global-local self-consistent field (SCF) iteration to determine local electronic wave functions & global electrostatic potential



- Globally scalable (real-space multigrid) + locally fast (plane wave) solver



F. Shimojo et al., *J. Chem. Phys.* **140**, 18A529 ('14)
K. Nomura et al., *IEEE/ACM Supercomputing, SC14* ('14)
cf. J. Lam et al., *Nature Commun.* **15**, 3479 ('24)