Parallel Quantum Dynamics

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Chemical Engineering & Materials Science
Department of Biological Sciences
University of Southern California

Email: anakano@usc.edu

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

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 by real-space multigrids; scalability behavior the same as 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(): ψj←exp(-iVjΔt/2)ψj (j∈[1,NX])
  kin_prop(Δt/2)
  kin_prop(Δt)
  kin_prop(Δt/2)
  pot_prop(): ψj←exp(-iVjΔt/2)ψj (j∈[1,NX])
```

http://cacs.usc.edu/education/phys516.html

$$\psi(t+\Delta t) \leftarrow \exp(-iV\Delta t/2)\exp(-iT_x\Delta t)\exp(-iV\Delta t/2)\psi(t)$$
$$= e^{-iV\Delta t/2} U_x^{(\mathrm{half})} U_x^{(\mathrm{full})} U_x^{(\mathrm{half})} e^{-iV\Delta t/2}\psi(t)$$

```
kin_prop(Δ)
  periodic_bc(): ψ0←ψNX; ψNX+1←ψ1
  for ∀j∈[1,NX]
    ψj←blx(Δ)jψj-1+al(Δ)jψj+bux(Δ)jψj+11
```

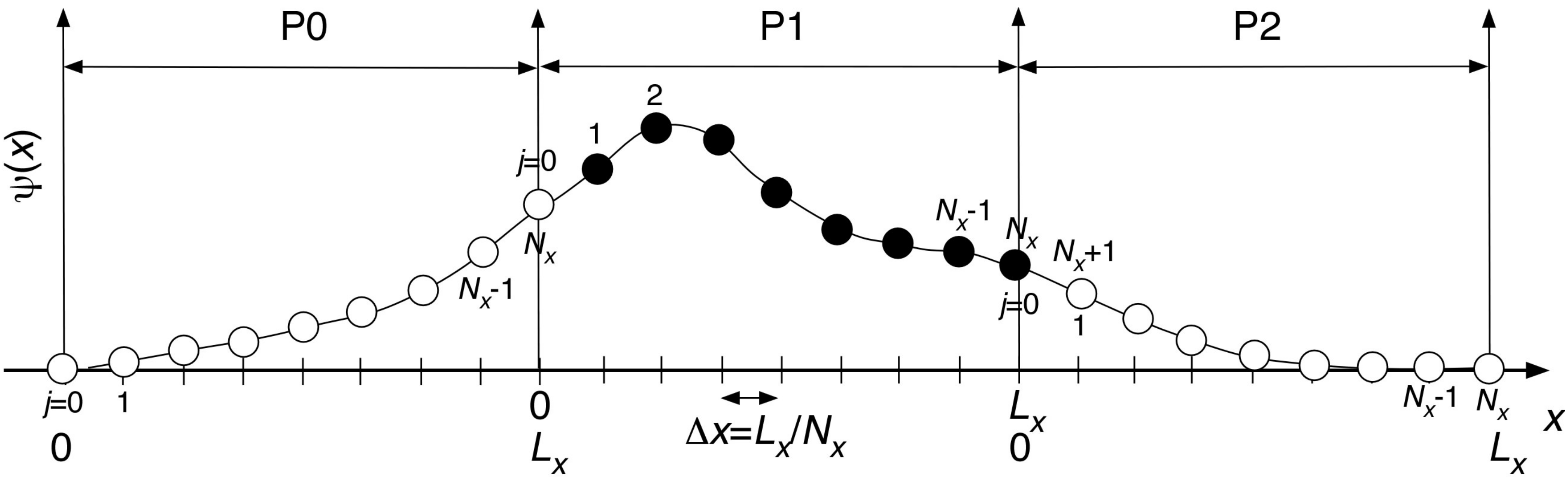
$$\begin{cases} \varepsilon_n^+ = \dfrac{1}{2}\left[\exp\left(-\dfrac{i\Delta t}{n}(a+b)\right) + \exp\left(-\dfrac{i\Delta t}{n}(a-b)\right)\right] \\ \varepsilon_n^- = \dfrac{1}{2}\left[\exp\left(-\dfrac{i\Delta t}{n}(a+b)\right) - \exp\left(-\dfrac{i\Delta t}{n}(a-b)\right)\right] \end{cases}$$

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

ψj(t+1)←f(ψj-1(t),ψj(t),ψj+1(t))(j∈[1,NX])

SC Parallelization

- **Self-centric spatial decomposition**



- **Local & global coordinates**

$$\begin{cases} x_j = j\Delta x \\ x_j^{(\mathrm{global})} = j\Delta x + \underbrace{pL_x}_{\text{off-set}} \end{cases}$$

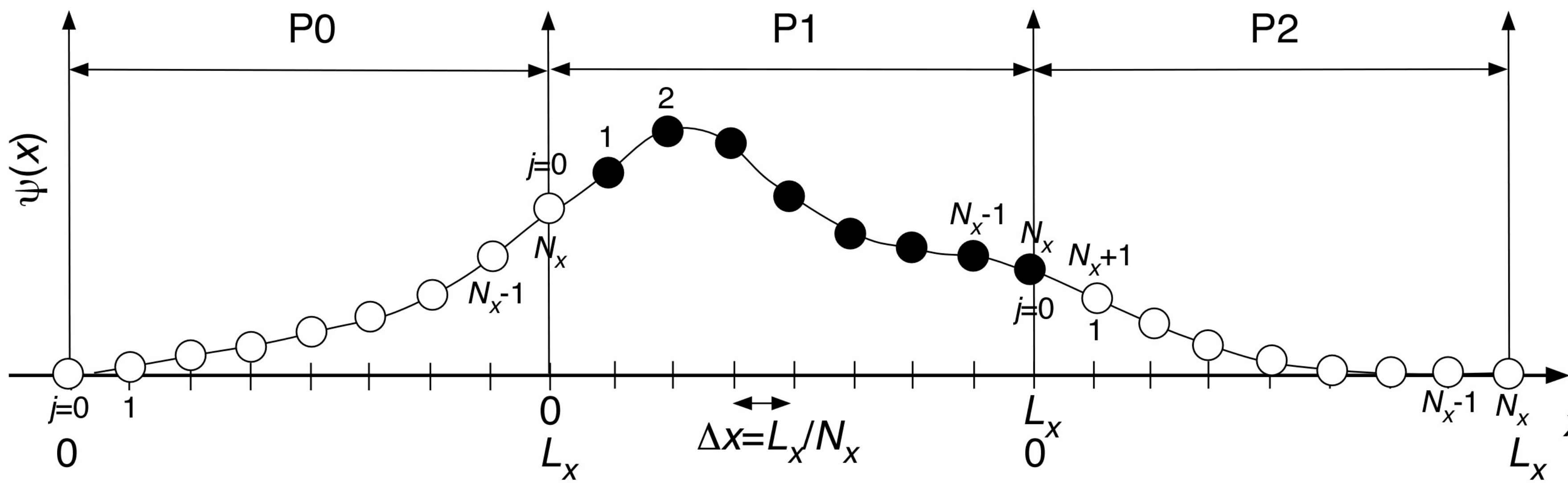
- **Global coordinates only in `init_prop()` & `init_wavefn()`**

http://cacs.usc.edu/education/cs653.html

Boundary Wave Function Caching

- **Parallelized `periodic_bc()`**

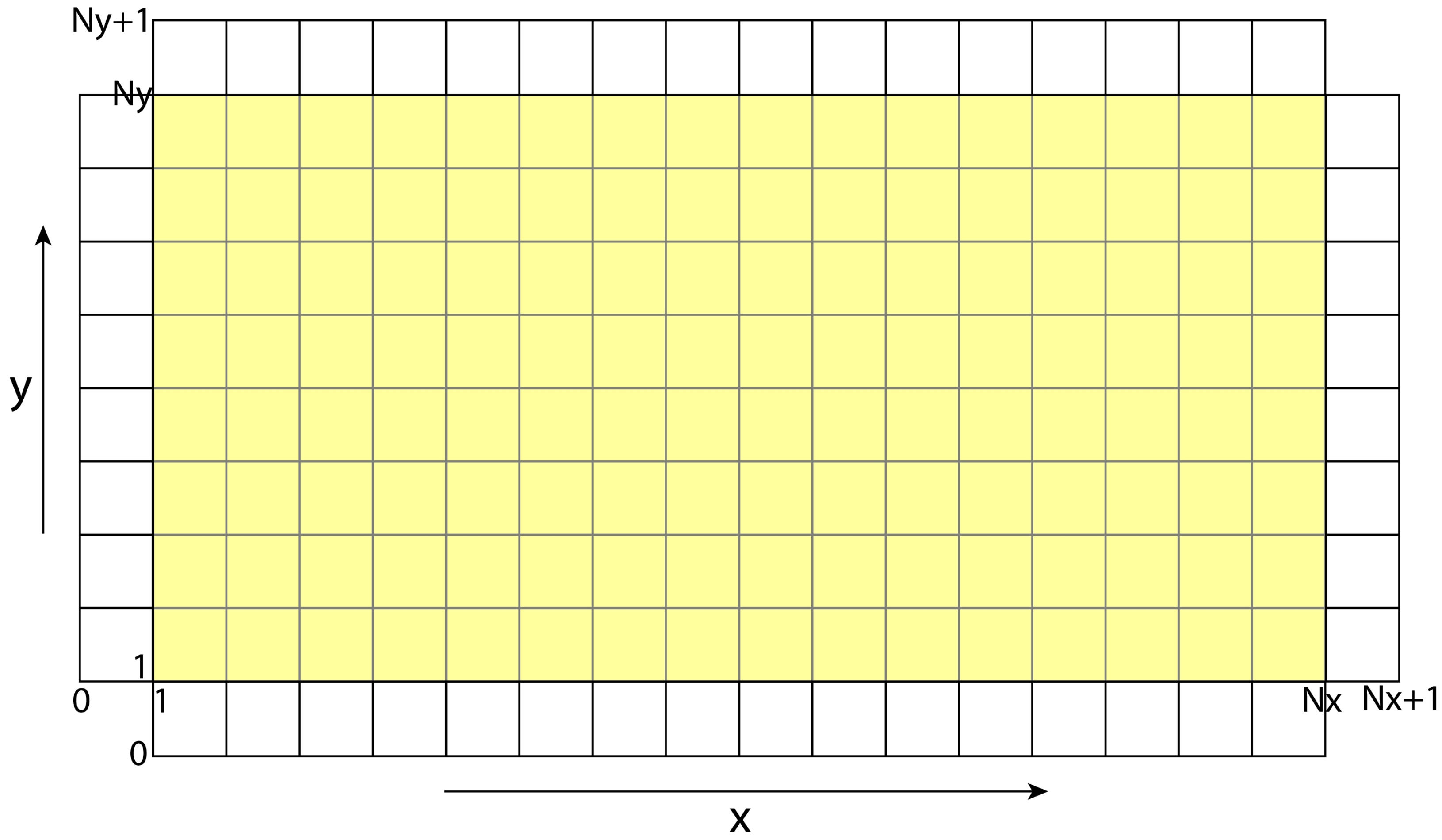
```
plw = (myid-1+nproc)%nproc; /* Lower partner process */
pup = (myid+1      )%nproc; /* Upper partner process */

/* 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];

/* 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 ∀directions
  send front row psi(...,1 or Nα,...) to forward neighbor
  receive back appendage psi(...,Nα+1 or 0,...) from back neighbor
```



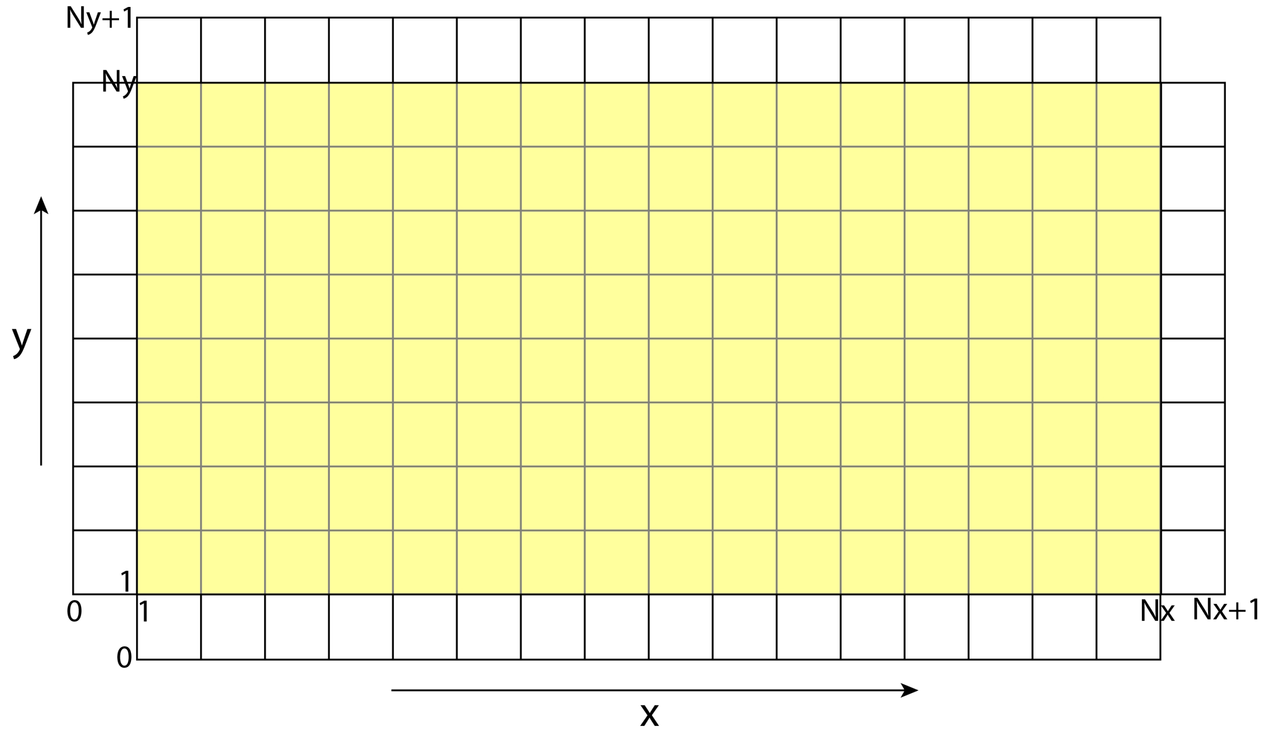
Multidimensional Parallelization

- **Message composition**

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

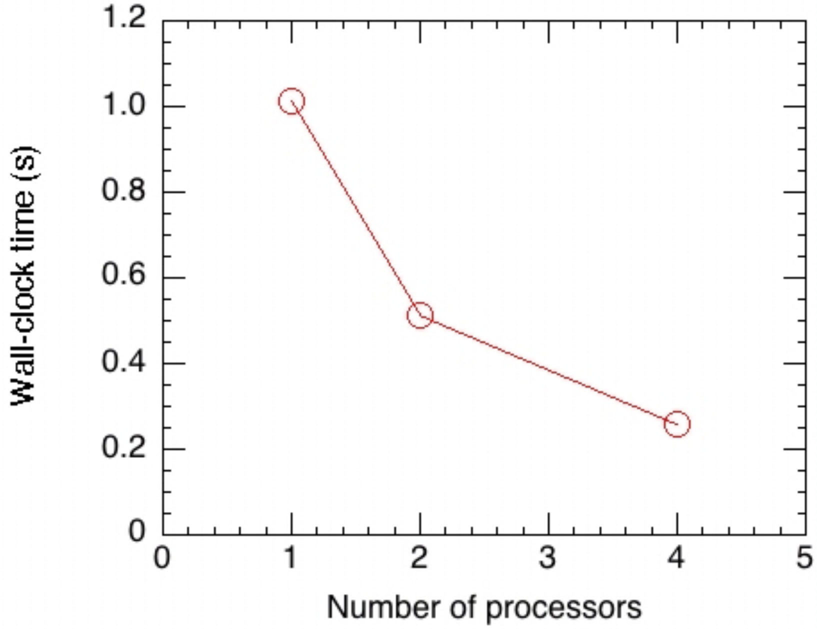
(Example) x-low direction

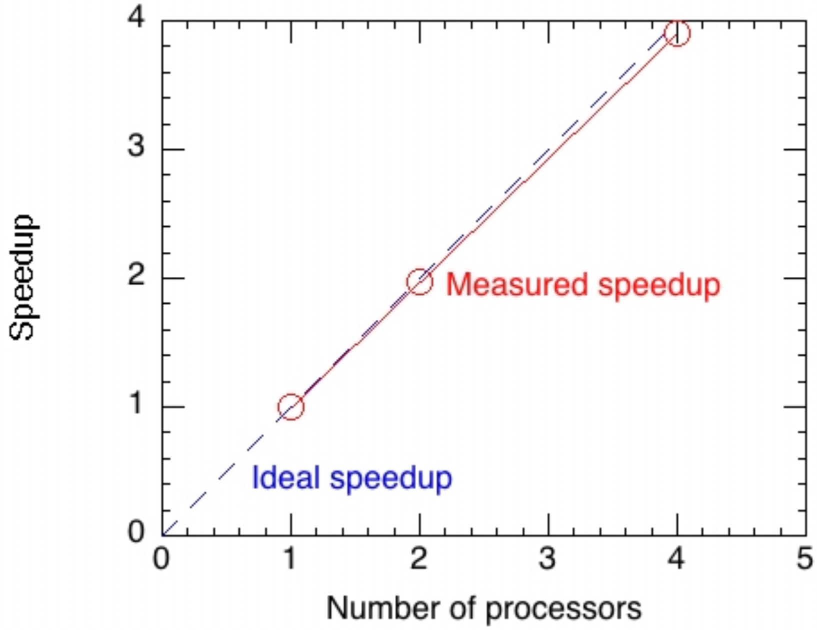
$$i_b = 1, i_e = 1, j_b = 1, j_e = N_y, k_b = 1, k_e = N_z$$
$$i'_b = N_x + 1, i'_e = N_x + 1, j'_b = 1, j'_e = N_y, k'_b = 1, k'_e = N_z$$

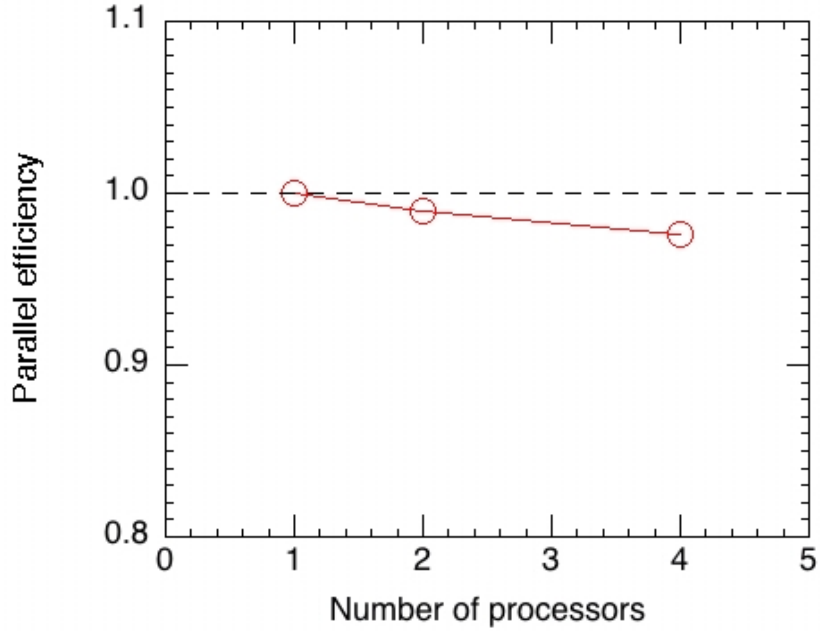


Parallel QD Results

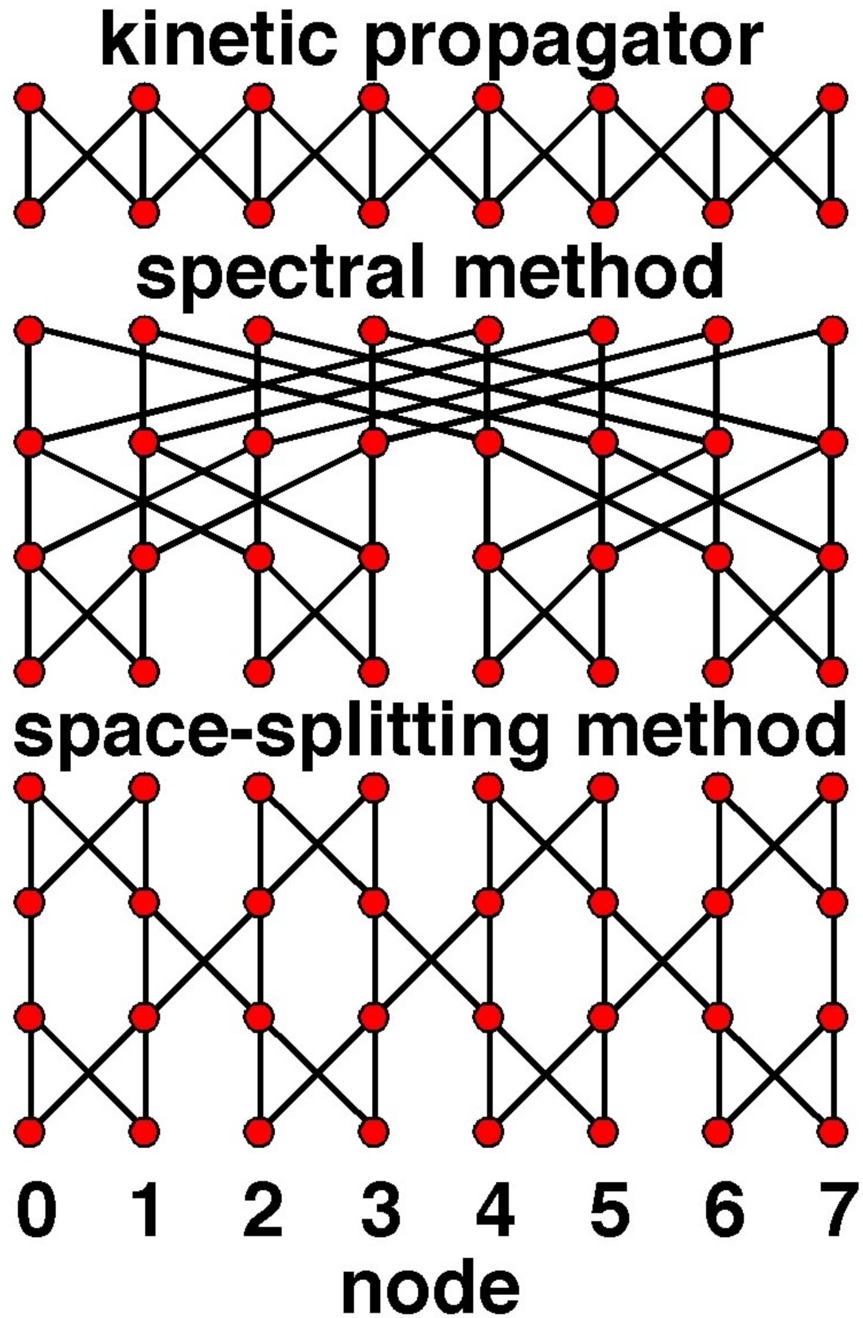
Strong scaling results







Parallel QD Communications



$\psi \leftarrow K\psi$

$\psi \leftarrow \exp(-iK\Delta)\psi$ — $O(MogN)$

$\psi \leftarrow \exp(-iK\Delta)\psi$ — $O(N)$

Nakano, *Comput. Phys. Commun.* **83**, 181 ('94)

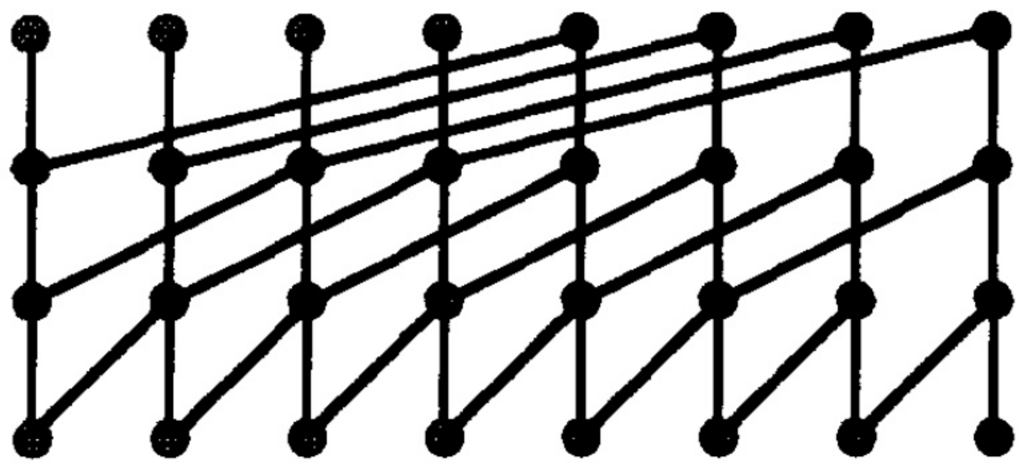
Parallel QD Algorithms

- **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**

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

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

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



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

Nakano, *Comput. Phys. Commun.* **83**, 181 ('94)