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 (easy spatial decomposition) 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 interaction by real-space multigrids; scalability behavior is similar to short-ranged (see slides 7-8 in** https://aiichironakano.github.io/cs653/02-04DC-slide.pdf**)**

S. C. Tiwari *et al*., *ACM HPCAsia2020*, Best Paper ('20)
F. Shimojo *et al*., *J. Chem. Phys.* **140**, 18A529 ('14)
K. Nomura *et al*., *IEEE/ACM Supercomputing, SC14* ('14)
A. Nakano, *Comput. Phys. Commun*. **83**, 181 ('94)

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

***cf.* velocity-Verlet half-time acceleration**

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

```
kin_prop($\Delta$)
  periodic_bc(): $\psi_0 \leftarrow \psi_{NX}$; $\psi_{NX+1} \leftarrow \psi_1$
  for $\forall j \in [1,NX]$
    $\psi_j \leftarrow \mathrm{blx}(\Delta)_j\psi_{j-1} + \mathrm{al}(\Delta)_j\psi_j + \mathrm{bux}(\Delta)_j\psi_{j+1}$
```

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

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

https://aiichironakano.github.io/phys516/03QD-slide.pdf

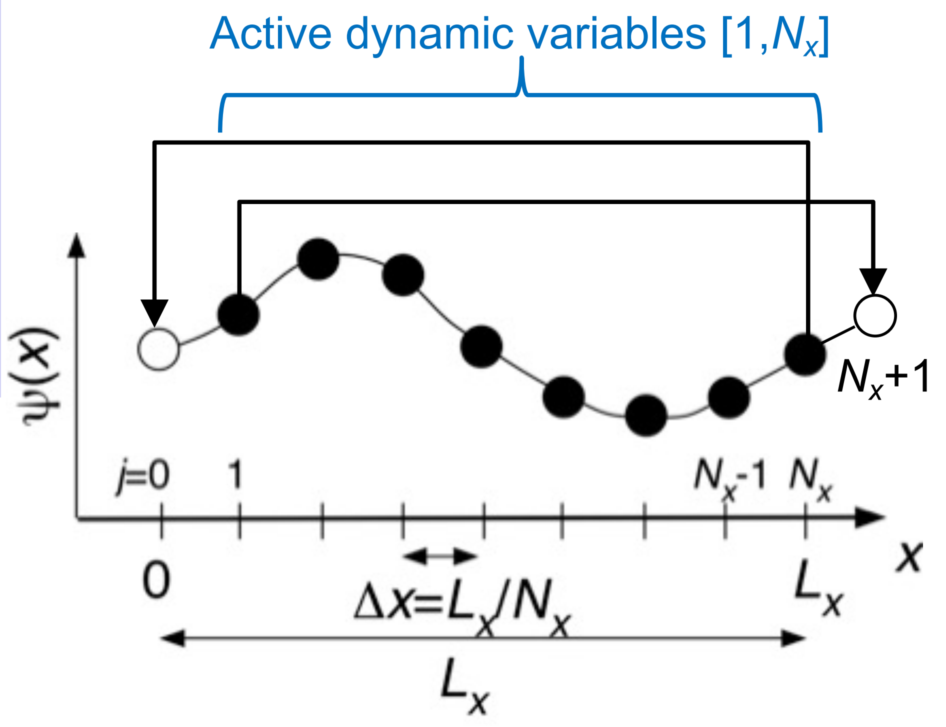
Quantum Dynamics Computation

- **Essence: Keep updating wave-function values mesh point-by-point as a function of those on the nearest-neighbor mesh points**

$$\psi_j(t+1) \leftarrow f(\psi_{j-1}(t), \psi_j(t), \psi_{j+1}(t)) \; (j \in [1, NX])$$

- **Periodic boundary condition *via* augmentation**

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

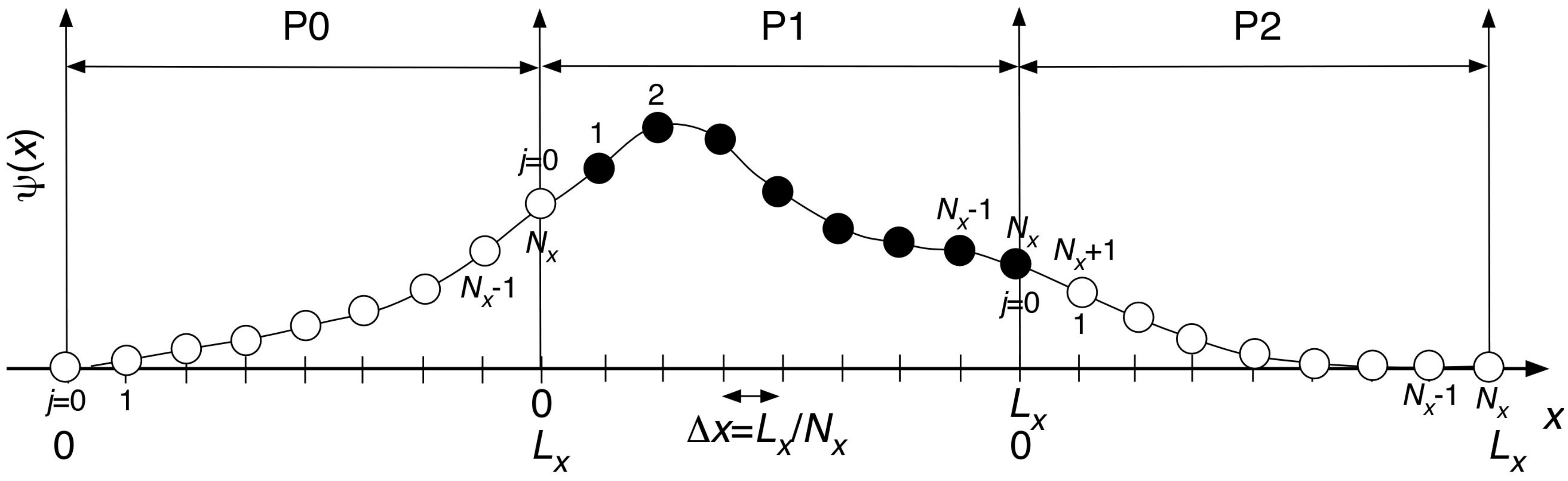


- **Often sufficient just to understand computational characteristics for parallelizing a serial code**

https://aiichironakano.github.io/cs596/src/qd

SC Parallelization

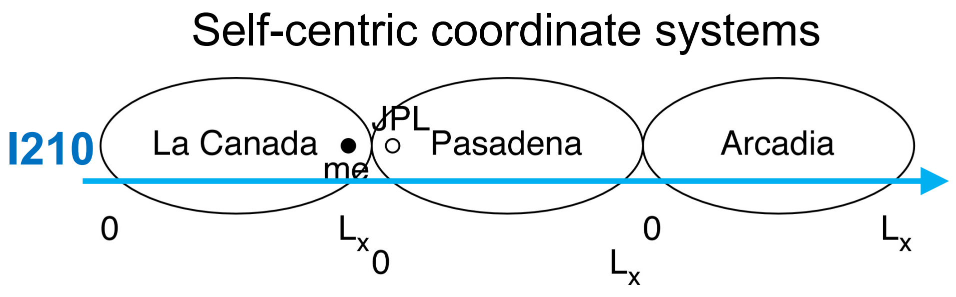
- **Self-centric spatial decomposition**



- **Local & global coordinates**

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

offset



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

Boundary Wave Function Caching

- **Parallelized `periodic_bc()`**

```
MPI_Comm_rank(MPI_COMM_WORLD, &myid);
MPI_Comm_size(MPI_COMM_WORLD, &nproc);
```

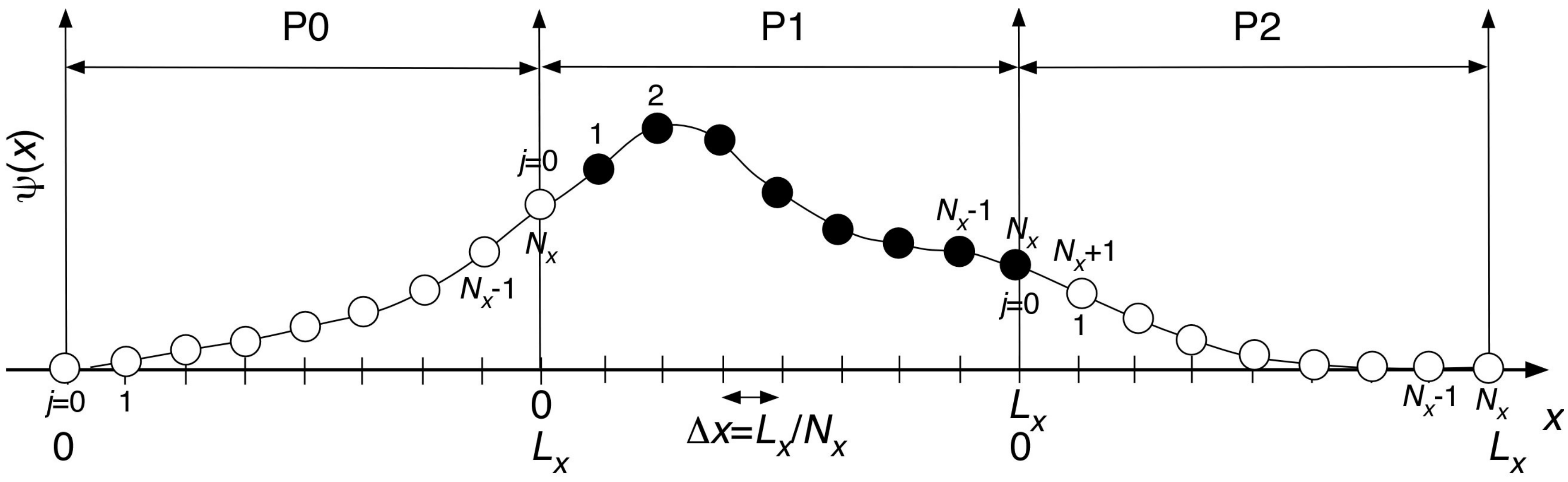
```
plw = (myid-1+nproc)%nproc; /* Lower partner process */
pup = (myid+1      )%nproc; /* Upper partner process */

/* 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];
```

I. Message (1D array) composition

II. Message passing

III. Message storing

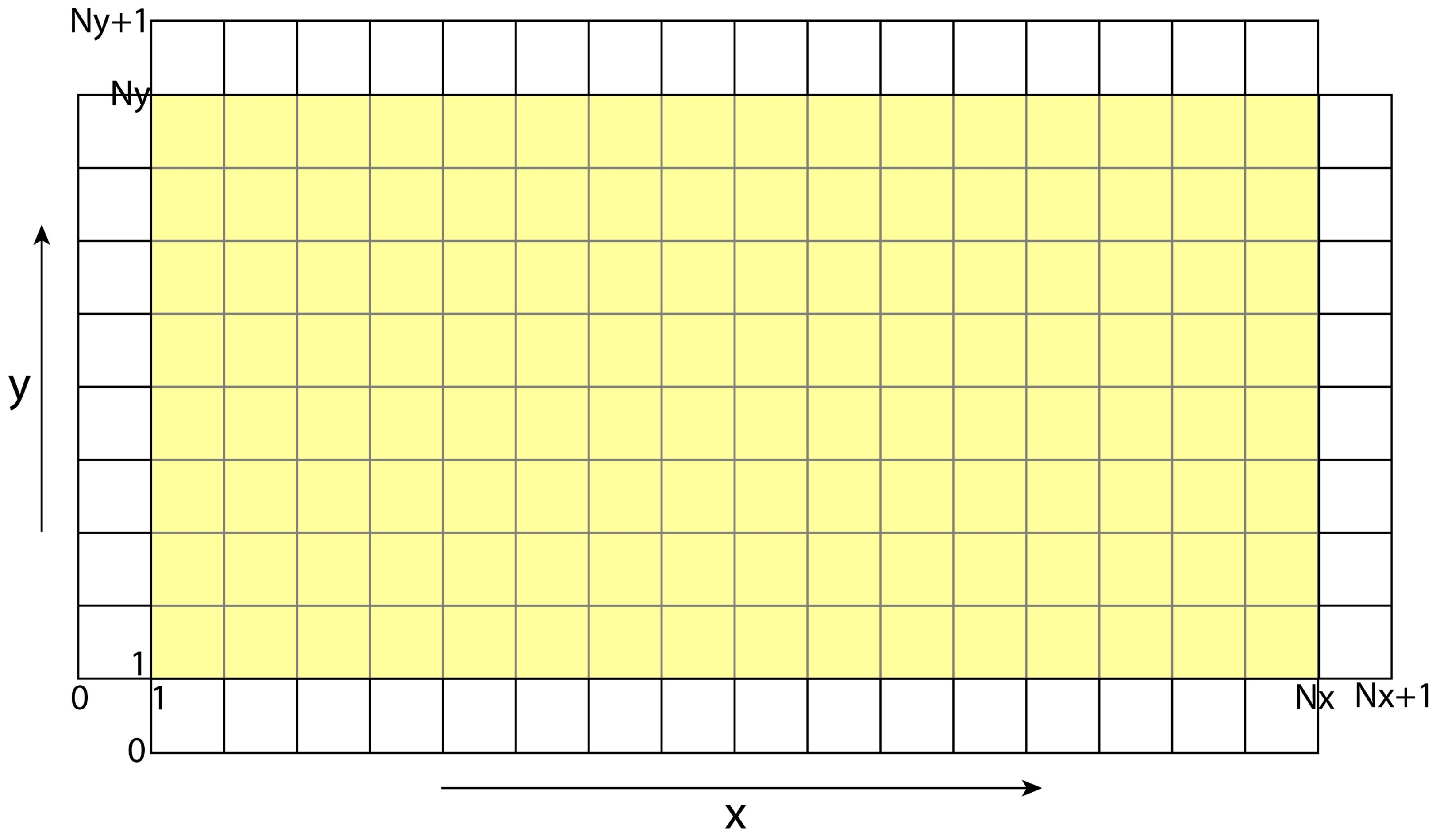
```
/* Cache boundary wave function value at the upper end */
dbuf[0:1] ← psi[1][0:1];
Send dbuf to plw;
Receive dbufr from pup;
psi[NX+1][0:1] ← dbufr[0:1];
```



Multidimensional Parallelization

- **Parallelized `periodic_bc()`**

```
for ∀directions
  send front row psi(...,1 or $N_\alpha$,...) to forward neighbor
  receive back appendage psi(...,$N_\alpha$+1 or 0,...) from back neighbor
```



Multidimensional Parallelization

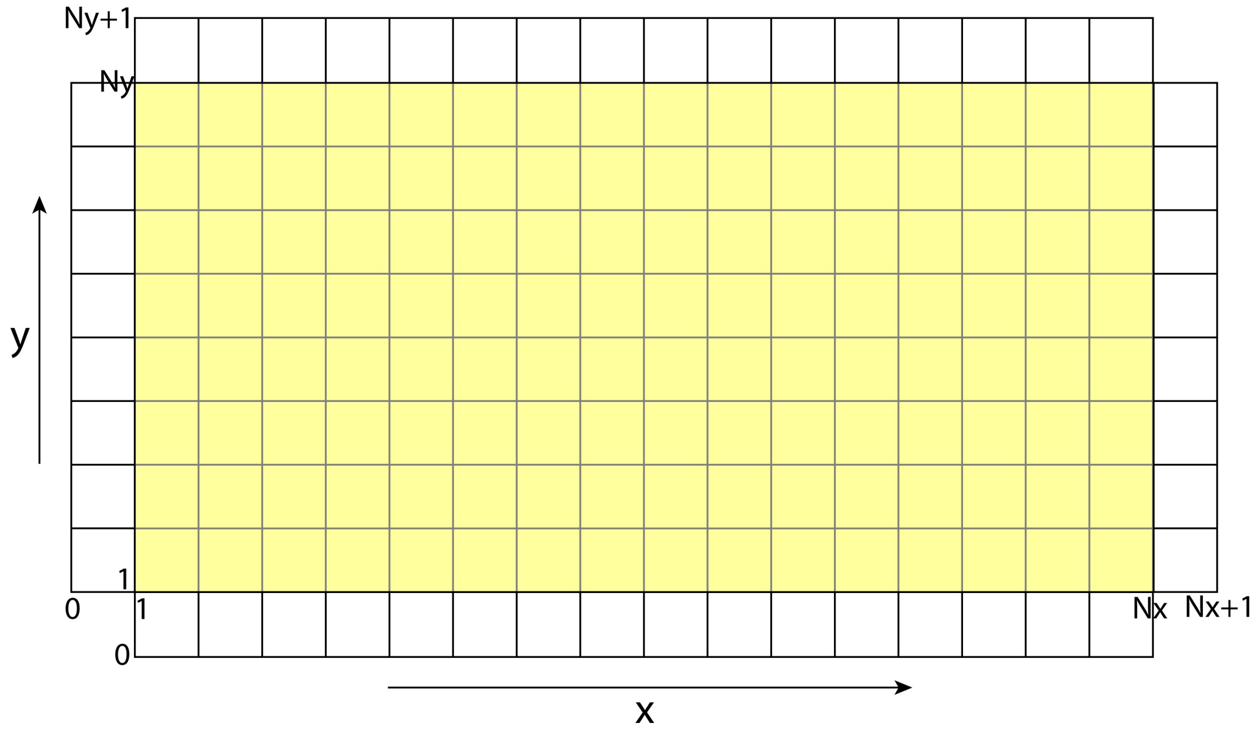
- **Message composition**

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

(Example) x-low direction

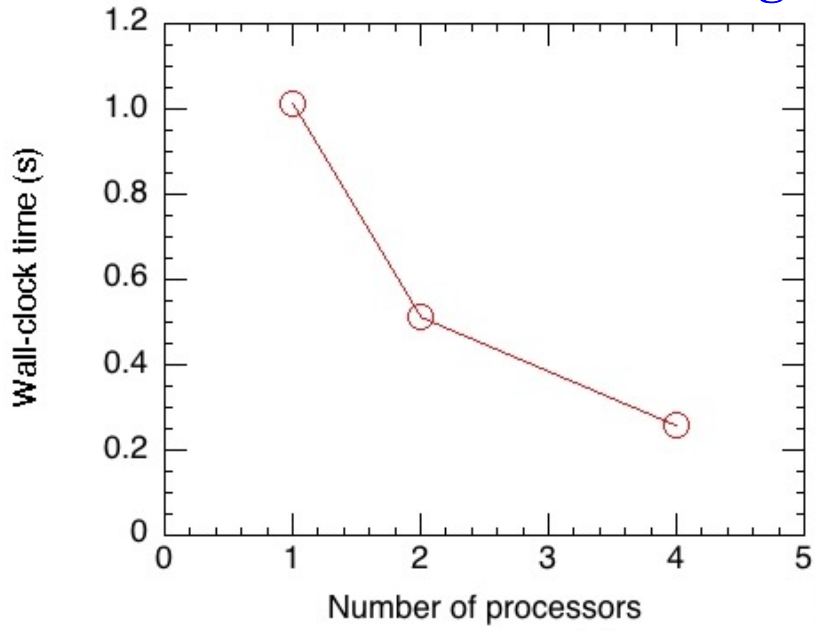
$\boldsymbol{i_b = 1, i_e = 1, j_b = 1, j_e = N_y, k_b = 1, k_e = N_z}$

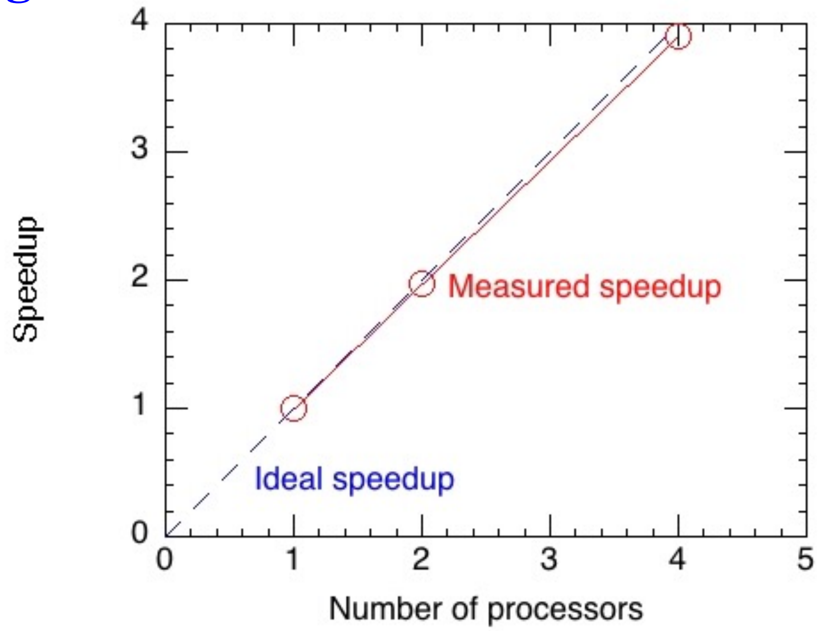
$\boldsymbol{i'_b = N_x+1, i'_e = N_x+1, j'_b = 1, j'_e = N_y, k'_b = 1, k'_e = N_z}$

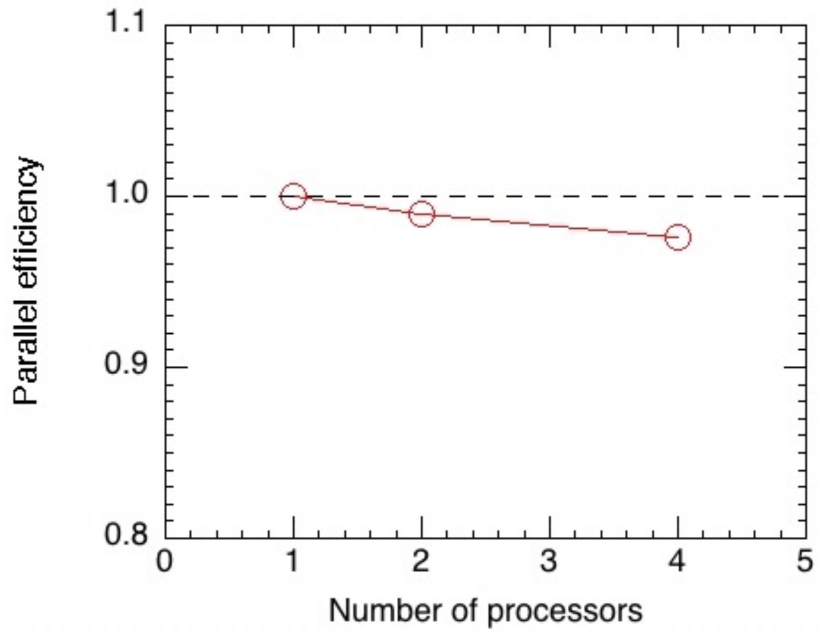


Parallel QD Results

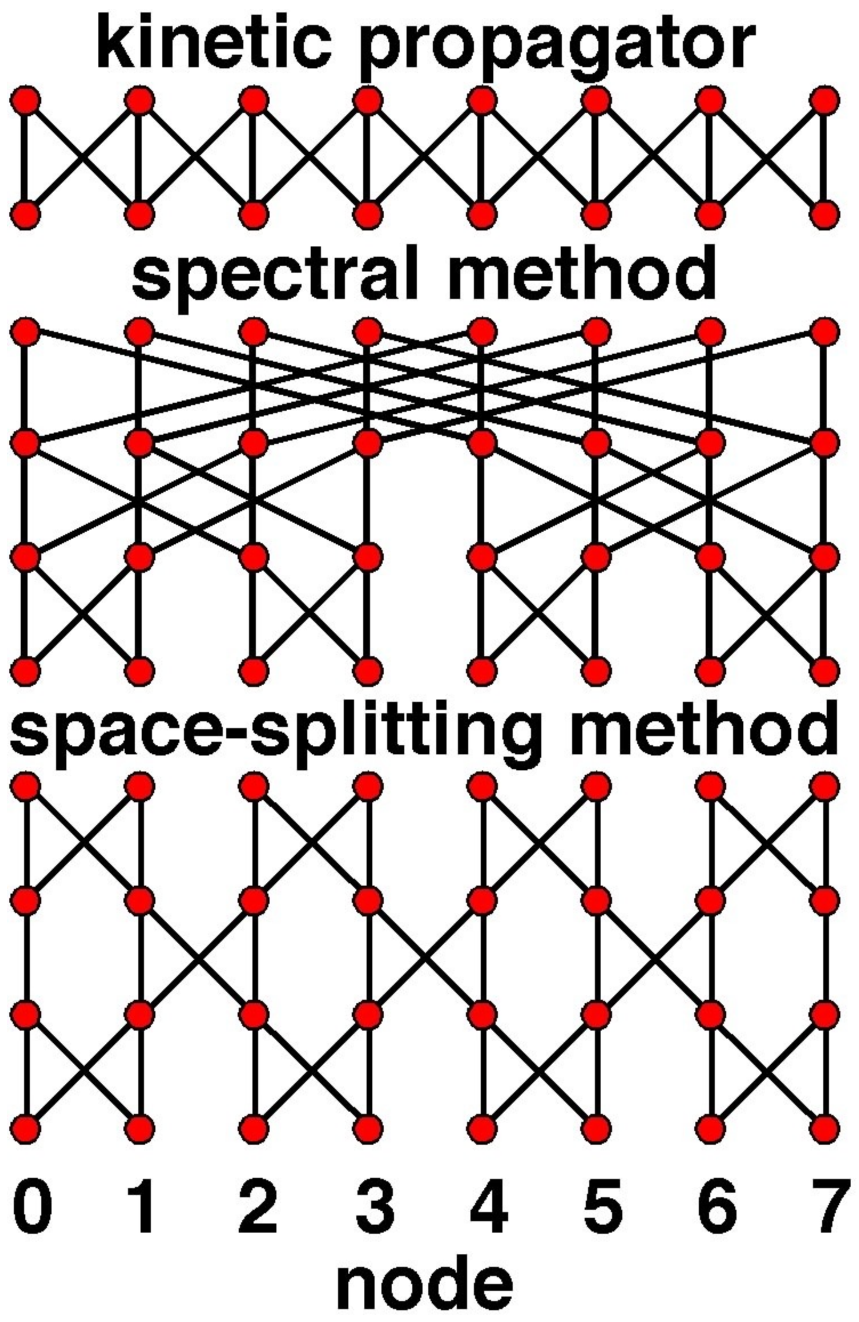
Strong scaling results







Parallel QD Communications



$\psi \leftarrow K\psi$

$\psi \leftarrow \exp(-iK\Delta)\psi$

O(*M*log*N*)

Up to 1,000 processors

$\psi \leftarrow \exp(-iK\Delta)\psi$

O(*N*)

Metascalable to future architectures

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**

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

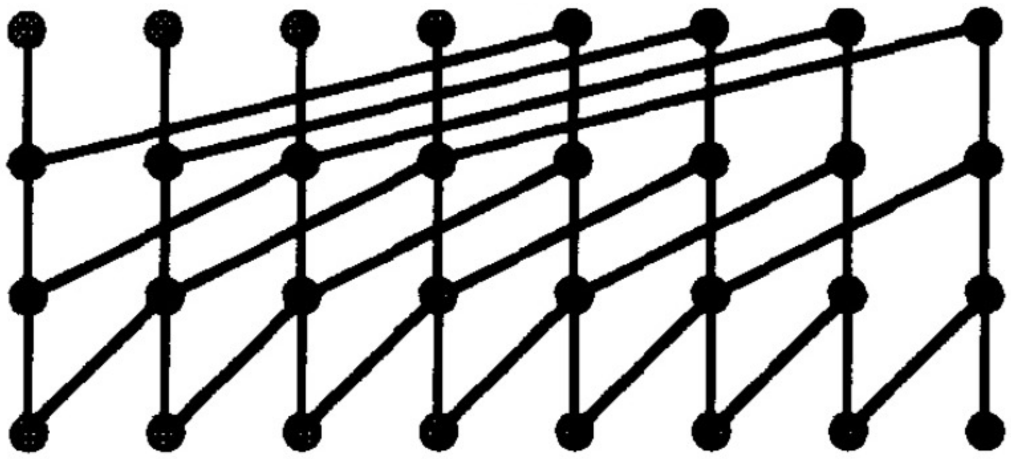
$$\underbrace{\left(1+\frac{i}{2\hbar}\hat{H}\Delta t\right)}_{A}\underbrace{\psi(t+\Delta t)}_{x} = \underbrace{\left(1-\frac{i}{2\hbar}\hat{H}\Delta t\right)\psi(t)}_{b}$$

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

$$\Longrightarrow$$

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

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



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