Introduction to 3-D finite element computation

Atsushi Suzuki

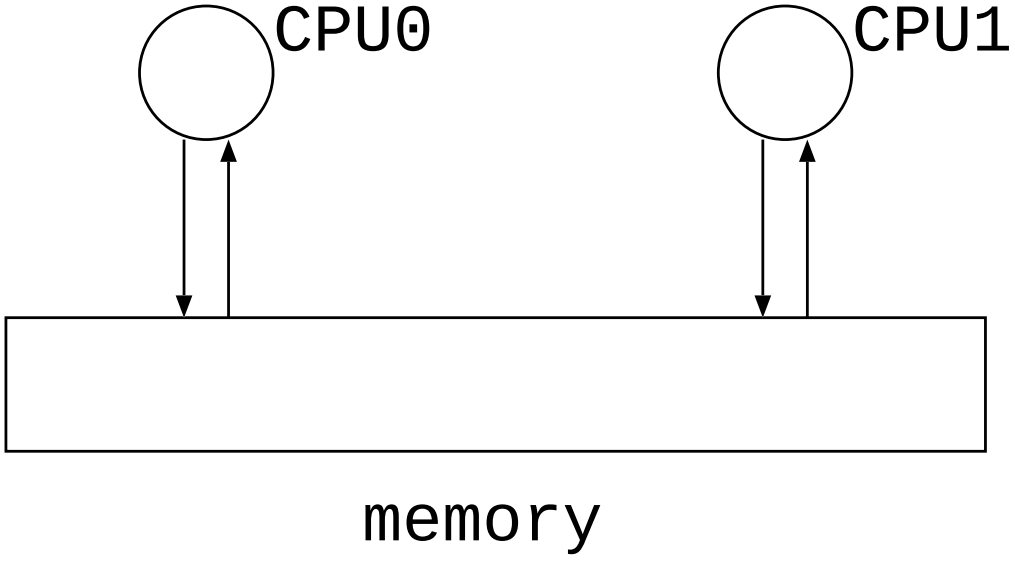
KM FJFI CVUT
Faculty of Mathematics, Kyushu University
asuzuki@math.kyushu-u.ac.jp

Parallel computation with domain decomposition.

- Architecture of parallel computers
  - shared memory type
  - distributed memory type
- Parallelization of matrix-vector product
  - non-overlapping domain decomposition
  - coarse grain parallelization with OpenMP
- Iterative substructuring method
  - Schur complement matrix
  - preconditioner

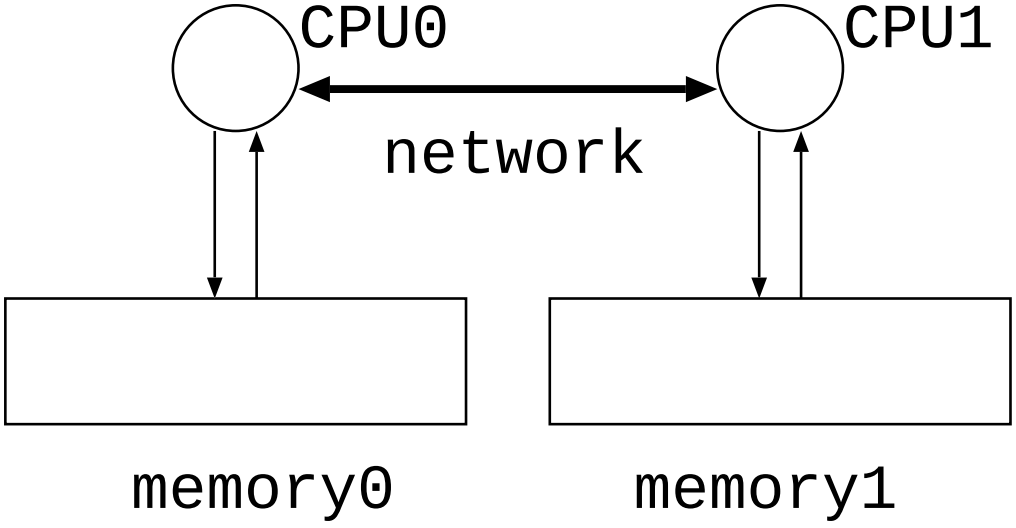
Architecture of parallel computers (1/4)

shared memory type:



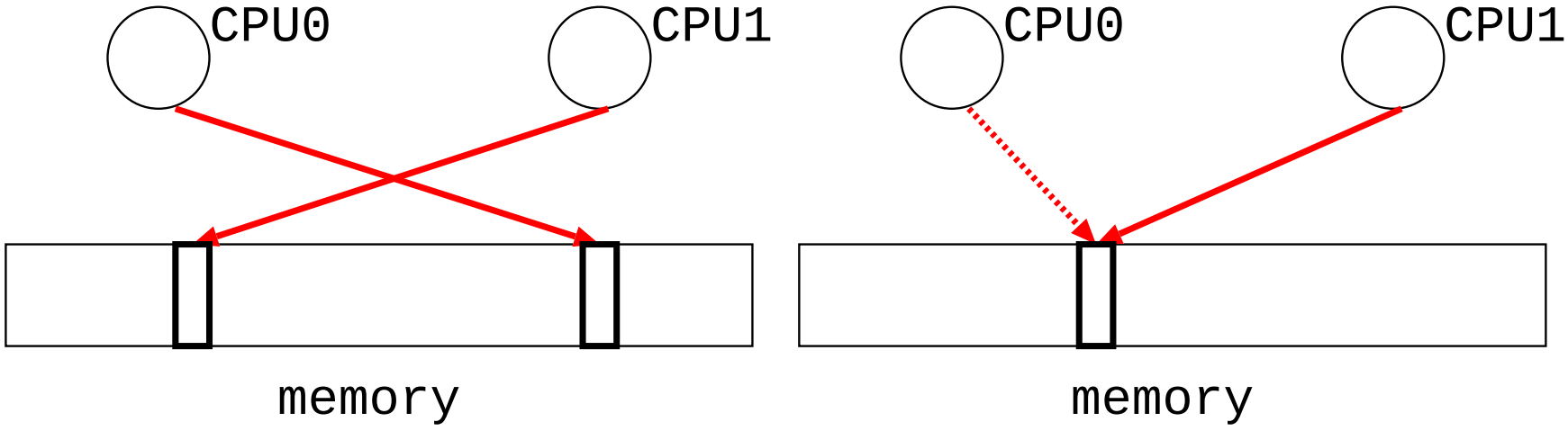
OpenMP

distributed memory type:



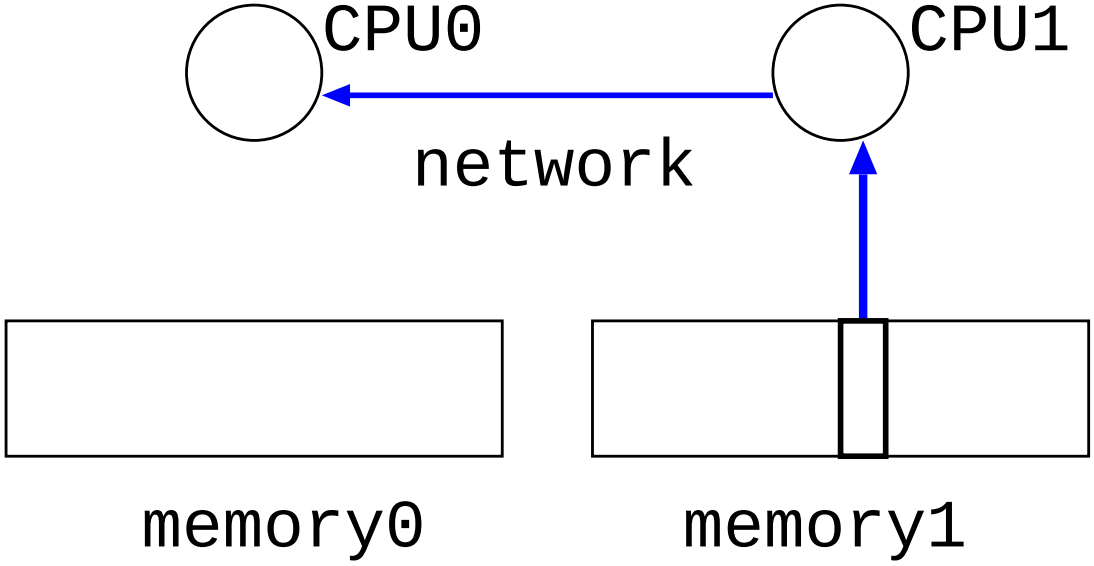
MPI

memory access on shared memory type:



2 CPUs can not write to same address simultaneously.

memory access on distributed memory type:

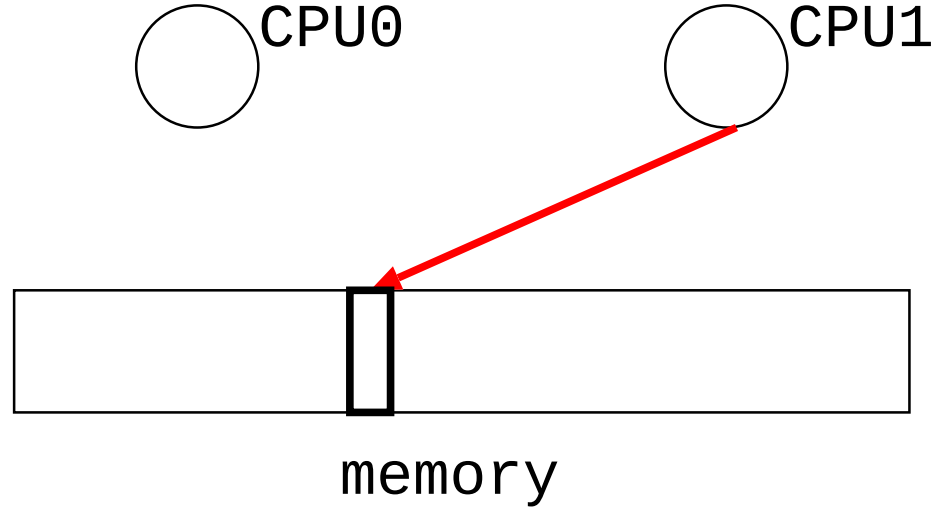


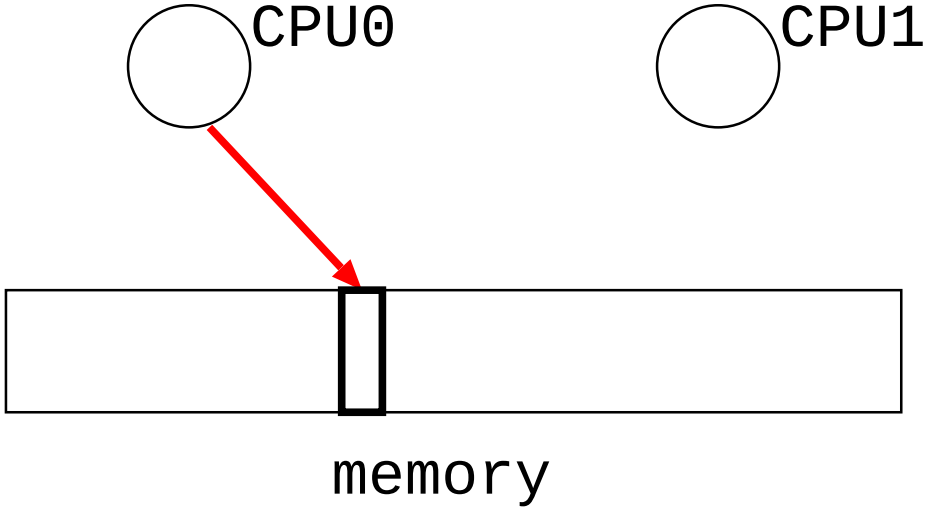
message passing

is required.

Architecture of parallel computers (3/4)

limitation on memory access on shared memory type:





control writing
to memory from CPU by
semaphore or barrier

Architecture of parallel computers (4/4)

Shared memory with OpenMP is better programming model.

- OpenMP is originally designed for loop parallelization (fine grain parallelization)
- Coarse grain parallelization is written by `parallel region`
- For large-scale parallel computer, Uniform Memory Access (UMA) architecture is not easy to develop and non-Uniform Memory Access (NUMA) is used.
- OpenMP has no capability to allocate memory in the way of NUMA architecture.

Parallelization of matrix-vector product (1/2)

$A : \mathbb{R}^{N\times N}$ sparse matrix, $x, b : \mathbb{R}^N$ vector.

linear equation $A\,x = b$

matrix-vector product $\Leftarrow$ Krylov subspace methods (CG, GMRES,...)

$$[y]_i := \sum_{1\le j\le N} [A]_{i\,j}[x]_j$$

$$[y]_i := \sum_{j\in\{j;[A]_{i\,j}\neq 0\}} [A]_{i\,j}[x]_j\,.$$

with CRS format:

$v_A[k] = A_{i\,j},\ \ c_A[k] = j,\ \ r_A[i] \le k < r_A[i+1],$

$$y[i] := \sum_{r_A[i]\le k<r_A[i+1]} v_A[k]x[c_A[k]]$$

Parallelization of matrix-vector product (2/2)

parallelization of loop by OpenMP:

```
#pragma omp parallel for private(j)
for (i = 0; i < N; i++) {
```

$$[v]_i := \sum_{j \in \{j; [A]_{i\,j} \neq 0\}} [A]_{i\,j}[u]_j \,;$$

```
}
```

loop counter variable j depends on loop counter variable i.

$\Rightarrow$ declaration of `private` is needed.

loop with length `N` is parallelized and executed by parallel processors

Block decomposition of matrix

non-overlapping decomposition of index set :

$\Lambda := \{1, 2, \ldots, N\}$

$\Lambda = \Lambda_1 \oplus \Lambda_2 \oplus \cdots \oplus \Lambda_D \oplus \Lambda_{\mathcal{F}}$ (direct sum)

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_D \\ y_{\mathcal{F}} \end{bmatrix} = \begin{bmatrix} A_{1\,1} & & & & A_{1\,\mathcal{F}} \\ & A_{2\,2} & & & A_{2\,\mathcal{F}} \\ & & \ddots & & \vdots \\ & & & A_{D\,D} & A_{D\,\mathcal{F}} \\ A_{\mathcal{F}\,1} & A_{\mathcal{F}\,2} & \cdots & A_{\mathcal{F}\,D} & A_{\mathcal{F}\,\mathcal{F}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \\ x_{\mathcal{F}} \end{bmatrix}$$

$[A_{p\,q}]_{i\,j} = 0 \quad (i \in \Lambda_p, j \in \Lambda_q), 1 \leq p < q \leq D$

$\leftarrow$ sparseness)

Block decomposition of matrix

non-overlapping decomposition of index set :

$\Lambda := \{1, 2, \ldots, N\}$

$\Lambda = \Lambda_1 \oplus \Lambda_2 \oplus \cdots \oplus \Lambda_D \oplus \Lambda_{\mathcal{F}}$ (direct sum)

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_D \\ y_{\mathcal{F}} \end{bmatrix} = \begin{bmatrix} A_{1\,1} & & & & A_{1\,\mathcal{F}} \\ & A_{2\,2} & & & A_{2\,\mathcal{F}} \\ & & \ddots & & \vdots \\ & & & A_{D\,D} & A_{D\,\mathcal{F}} \\ A_{\mathcal{F}\,1} & A_{\mathcal{F}\,2} & \cdots & A_{\mathcal{F}\,D} & A_{\mathcal{F}\,\mathcal{F}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \\ x_{\mathcal{F}} \end{bmatrix}$$

$[A_{p\,q}]_{i\,j} = 0 \quad (i \in \Lambda_p, j \in \Lambda_q), 1 \leq p < q \leq D$

decomposition of index $\Lambda_{\mathcal{F}}$

$$\Lambda_{\mathcal{F}}^{(p)} := \{j \in \Lambda_{\mathcal{F}}\,;\, [A_{p\,\mathcal{F}}]_{i\,j} \neq 0 \quad \forall i \in \Lambda_p\}$$

Block decomposition of matrix

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_D \\ y_{\mathcal{F}} \end{bmatrix} = \begin{bmatrix} A_{1\,1} & & & & A_{1\,\mathcal{F}} \\ & A_{2\,2} & & & A_{2\,\mathcal{F}} \\ & & \ddots & & \vdots \\ & & & A_{D\,D} & A_{D\,\mathcal{F}} \\ A_{\mathcal{F}\,1} & A_{\mathcal{F}\,2} & \cdots & A_{\mathcal{F}\,D} & A_{\mathcal{F}\,\mathcal{F}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \\ x_{\mathcal{F}} \end{bmatrix}$$

decomposition of index $\Lambda_{\mathcal{F}}$

$$\Lambda_{\mathcal{F}}^{(p)} := \{ j \in \Lambda_{\mathcal{F}} \,;\, [A_{p\,\mathcal{F}}]_{i\,j} \neq 0 \quad \forall i \in \Lambda_p \}$$

$A_{\mathcal{F}\,\mathcal{F}} = \sum_{1 \leq p \leq D} A_{\mathcal{F}\,\mathcal{F}}^{(p)}$

$$A\,x = \sum_{1 \leq p \leq D} \begin{bmatrix} A_{p\,p} & A_{p\,\mathcal{F}} \\ A_{\mathcal{F}\,p} & A_{\mathcal{F}\,\mathcal{F}}^{(p)} \end{bmatrix} \begin{bmatrix} x_p \\ x_{\mathcal{F}}^{(p)} \end{bmatrix}$$

Block matrix by non-overlapping domain decomposition

A: stiffness matrix

non-overlapping domain decomposition:

D subdomains + interface

$\Omega_p (1 \le p \le D), \quad \mathcal{F} := \bigcup_{1 \le p < q \le D} \partial\Omega_p \cap \partial\Omega_q.$

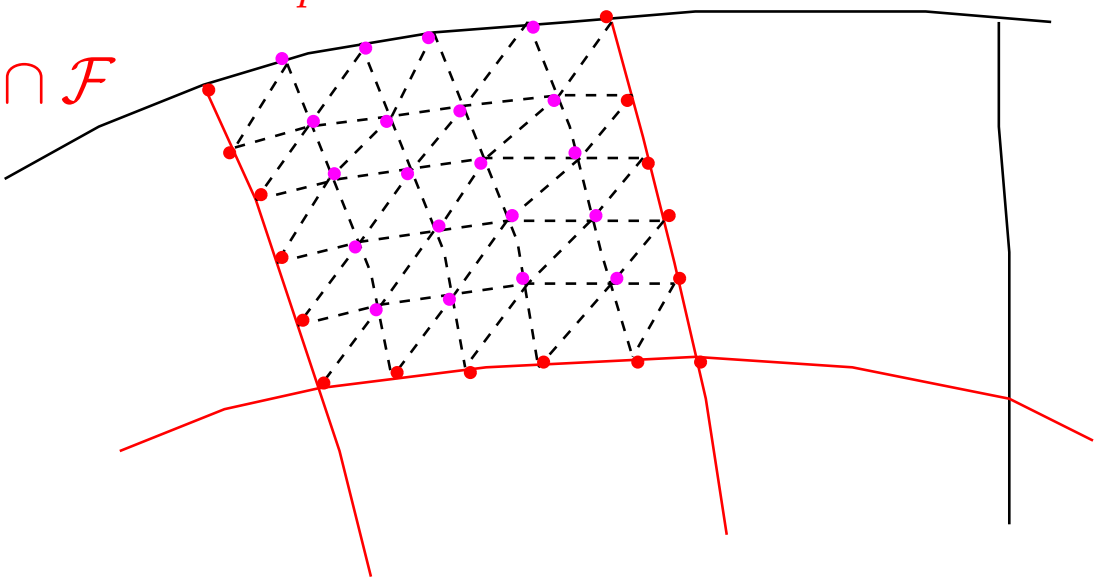
$\bar{\Omega} = \bigcup_{1 \le p \le D} (\bar{\Omega}_p \setminus \mathcal{F}) \cup \mathcal{F}\,.$

A_{pp} nodes on $\bar{\Omega}_p \setminus \mathcal{F}$

$A_{p\mathcal{F}}$ nodes on $\bar{\Omega}_p \setminus \mathcal{F}$ and $\partial\Omega_p \cap \mathcal{F}$

$A^{(p)}_{\mathcal{F}\mathcal{F}}$ nodes on $\partial\Omega_p \cap \mathcal{F}$

$$\sum_{1 \le p < D} \begin{bmatrix} A_{pp} & A_{p\mathcal{F}} \\ A_{\mathcal{F}p} & A^{(p)}_{\mathcal{F}\mathcal{F}} \end{bmatrix}$$



Parallel computation of matrix-vector product

Step 1

allocate working array $y_p, y_{\mathcal{F}}^{(p)}$.

Step 2

matrix-vector product by each processor p:

$$\begin{bmatrix} y_p \\ y_{\mathcal{F}}^{(p)} \end{bmatrix} = \begin{bmatrix} A_{p\,p} & A_{p\,\mathcal{F}} \\ A_{\mathcal{F}\,p} & A_{\mathcal{F}\,\mathcal{F}}^{(p)} \end{bmatrix} \begin{bmatrix} x_p \\ x_{\mathcal{F}}^{(p)} \end{bmatrix} .$$

Step 3

sum operation with index Λ_D:

$$y_{\mathcal{F}} = \sum_{1 \le q \le D} y_{\mathcal{F}}^{(q)} .$$

Parallel computation of matrix-vector product

Step 1

allocate working array $y_p, y_{\mathcal{F}}^{(p)}$.

Step 2

matrix-vector product by each processor p:

$$\begin{bmatrix} y_p \\ y_{\mathcal{F}}^{(p)} \end{bmatrix} = \begin{bmatrix} A_{p\,p} & A_{p\,\mathcal{F}} \\ A_{\mathcal{F}\,p} & A_{\mathcal{F}\,\mathcal{F}}^{(p)} \end{bmatrix} \begin{bmatrix} x_p \\ x_{\mathcal{F}}^{(p)} \end{bmatrix} .$$

direct sum $\Lambda_{\mathcal{F}} = \Lambda_{\mathcal{F},1} \oplus \Lambda_{\mathcal{F},2} \oplus \cdots \oplus \Lambda_{\mathcal{F},D}$:

Step 3'

sum operation by each processor p:

$$[y_{\mathcal{F}}]_j = \sum_{1 \le q \le D} [y_{\mathcal{F}}^{(q)}]_j \quad j \in \Lambda_{\mathcal{F},p} .$$

Parallel computation of matrix-vector product

Step 1

allocate working array $y_p, y_{\mathcal{F}}^{(p)}$.

Step 2

matrix-vector product by each processor p:

$$\begin{bmatrix} y_p \\ y_{\mathcal{F}}^{(p)} \end{bmatrix} = \begin{bmatrix} A_{p\,p} & A_{p\,\mathcal{F}} \\ A_{\mathcal{F}\,p} & A_{\mathcal{F}\,\mathcal{F}}^{(p)} \end{bmatrix} \begin{bmatrix} x_p \\ x_{\mathcal{F}}^{(p)} \end{bmatrix} .$$

wait until completion of step 2: $\Leftrightarrow$ `barrier`

Step 3'

sum operation by each processor p:

$$[y_{\mathcal{F}}]_j = \sum_{1 \le q \le D} [y_{\mathcal{F}}^{(q)}]_j \quad j \in \Lambda_{\mathcal{F},p} .$$

Implementation by OpenMP with orphaned directive

```
#include <omp.h>
```
allocate $x_{\mathcal{F}}$, $y_{\mathcal{F}}$;
```
omp_set_num_threads(D);
#pragma omp parallel default(shared) private (p)
{
```
  p=omp_get_thread_num();
  allocate x_p, y_p, $x_{\mathcal{F}}^{(p)}$, $y_{\mathcal{F}}^{(p)}$;
  copy data from $x_{\mathcal{F}}$ to $x_{\mathcal{F}}^{(p)}$;

$$\begin{bmatrix} y_p \\ y_{\mathcal{F}}^{(p)} \end{bmatrix} = \begin{bmatrix} A_{p\,p} & A_{p\,\mathcal{F}} \\ A_{\mathcal{F}\,p} & A_{\mathcal{F}\,\mathcal{F}}^{(p)} \end{bmatrix} \begin{bmatrix} x_p \\ x_{\mathcal{F}}^{(p)} \end{bmatrix} ;$$

```
#pragma omp barrier
```
  $[y_{\mathcal{F}}]_j = \sum_{1 \le q \le D} [y_{\mathcal{F}}^{(p)}]_j \quad j \in \Lambda_{\mathcal{F},p}$;
```
}
```

Implementation by OpenMP with orphaned directive

```
#include <omp.h>
allocate x_F, y_F;
omp_set_num_threads(D);
#pragma omp parallel default(shared) private (p)
{
  p=omp_get_thread_num();
  allocate x_p, y_p, x_F^(p), y_F^(p);
  copy data from x_F to x_F^(p);
```

$$\begin{bmatrix} y_p \\ y_{\mathcal{F}}^{(p)} \end{bmatrix} = \begin{bmatrix} A_{pp} & A_{p\mathcal{F}} \\ A_{\mathcal{F}p} & A_{\mathcal{F}\mathcal{F}}^{(p)} \end{bmatrix} \begin{bmatrix} x_p \\ x_{\mathcal{F}}^{(p)} \end{bmatrix} ;$$

```
#pragma omp barrier
```

$$[y_{\mathcal{F}}]_j = \sum_{1 \le q \le D} [y_{\mathcal{F}}^{(p)}]_j \quad j \in \Lambda_{\mathcal{F},p} ;$$

```
}
```

- first touch policy: memory allocated in parallel region for NUMA architecture
- threads are not joined nor generated. barrier is only used.

Parallel performance on IBM System p5/595 (1/2)

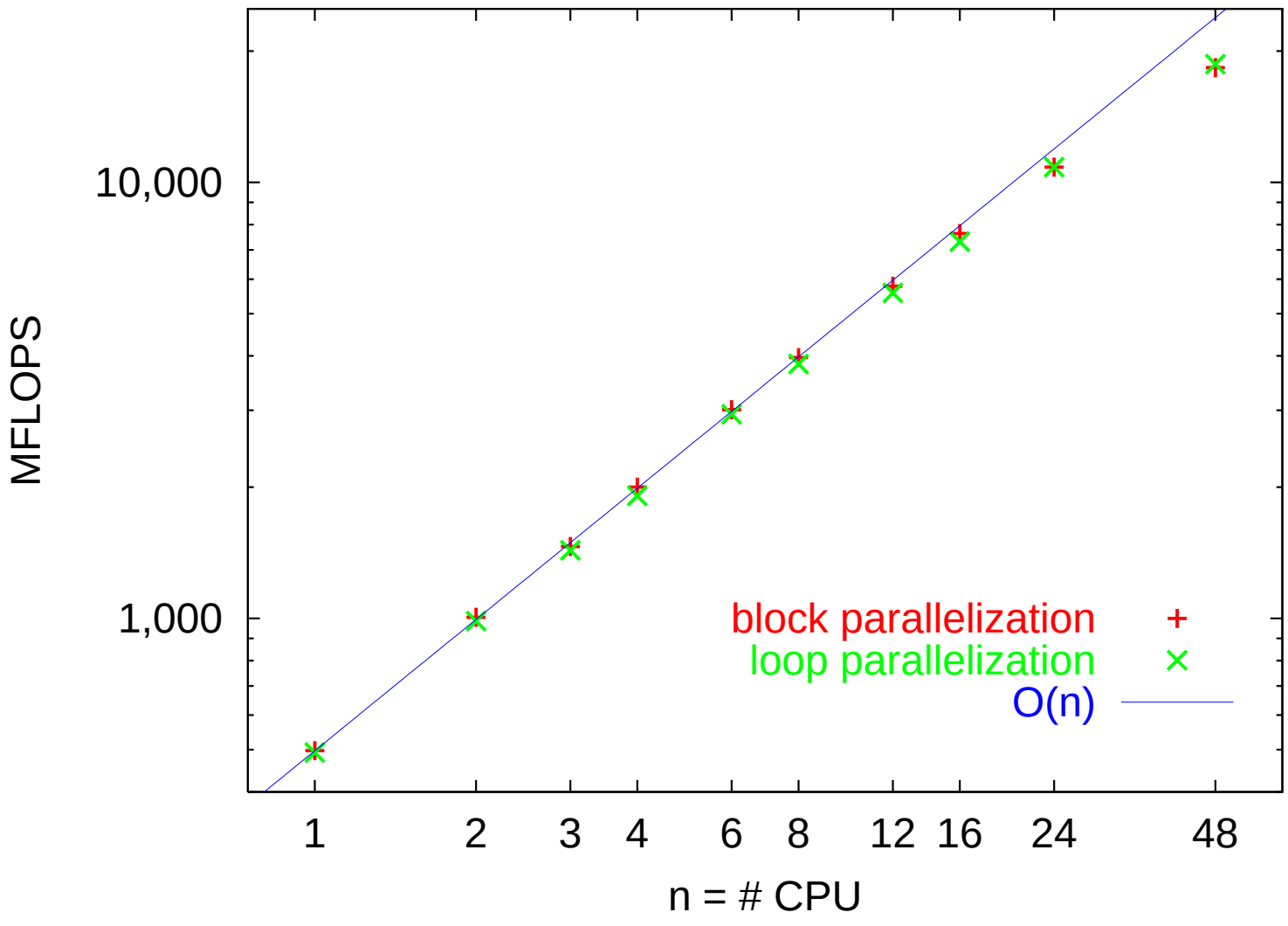
N	nonzeros	operations (MFLOP)	
		block parallel	loop parallel
22,932,588	524,436,522	2,124.846	2,074.813

	block parallel		loop parallel	
# CPU	time (sec.)	MFLOPS	time (sec.)	MFLOPS
1	4.27348	497.22	4.21108	492.70
2	2.11207	1,006.05	2.10430	985.99
3	1.45398	1,461.40	1.44805	1,432.83
4	1.06202	2,000.75	1.08669	1,909.29
6	0.706038	3,009.53	0.705740	2,939.92
8	0.535862	3,965.28	0.541420	3,832.20
12	0.367691	5,778.89	0.372019	5,577.17
16	0.278528	7,628.83	0.283959	7,306.78
24	0.196061	10,837.67	0.191501	10,835.97
48	0.115946	18,326.17	0.111125	18,670.98

number of subdomains is fixed as $D = 48$.

IBM System p5/595, Power5+ @ 2.1GHz $\times$ 64, SMP

Parallel performance on IBM System p5/595 (2/2)



Implementation of inner product by OpenMP

inner product of (x, y).

```
#include <omp.h>
```

allocate s[D], σ[D];

```
omp_set_num_threads(D);
#pragma omp parallel default(shared) private (p)
{
```

  p=omp_get_thread_num();

  s[p]=$(x_p, y_p) + \sum_{j \in \Lambda_{\mathcal{F},p}} [x_{\mathcal{F}}]_j [y_{\mathcal{F}}]_j$.

```
#pragma omp barrier
```

  σ[p] = $\sum_{1 \le q \le D}$ s[q] .

```
}
```

Parallel implementation of CG method by OpenMP

```
#pragma omp parallel default(shared) private ...{
```

$x_0 \in \mathbb{R}^N$,

```
#pragma omp barrier
```

$w_0 := Ax_0$,

$p_0 := r_0 := b - w_0$,

$\gamma_0 := (r_0,\, r_0)$,

do $n = 0, 1, \ldots,$ until $||r_n||/||r_0|| < \varepsilon$

```
#pragma omp barrier
```

$w_n := Ap_n$,

$\rho_n := (w_n,\, p_n)$,

$x_{n+1} := x_n + (\gamma_n/\rho_n)p_n$,

$r_{n+1} := r_n - (\gamma_n/\rho_n)w_n$,

$\gamma_{n+1} := (r_{n+1},\, r_{n+1})$,

$p_{n+1} := r_{n+1} + (\gamma_{n+1}/\gamma_n)p_n$.

enddo

}

matrix-vector products and inner product contain barrier.

Non-overlapping domain decomposition method

- Schur complement
  - iterative substructuring method
  - balancing Neumann-Neumann proconditioner

- Lagrange multiplier
  - FETI (finite element tearing and interconnecting)
  - mortar method (subdomains are discretized with different manner, e.g. different mesh sizes, different way of discretizations)

Iterative substructuring method (1/2)

non-overlapping decompositions of domain and matrix

$\overline{\Omega} = \bigcup_{1\le p\le D} \overline{\Omega^{(p)}} = \mathcal{F} \cup \bigcup_{1\le p\le D} (\partial\Omega \cap \Omega^{(p)})$,
$\mathcal{F} := \bigcup_{1\le p<q\le D} \partial\Omega^{(p)} \cap \partial\Omega^{(q)}$.

$$A = \sum_{1\le p\le D} \tilde{\mathcal{R}}^{(p)T} \begin{bmatrix} A_{pp} & A_{p\mathcal{F}} \\ A_{\mathcal{F}p} & A^{(p)}_{\mathcal{F}\mathcal{F}} \end{bmatrix} \tilde{\mathcal{R}}^{(p)} \qquad : \text{SPD}$$

local Schur complement matrix

$$S^{(p)} = A^{(p)}_{\mathcal{F}\mathcal{F}} - A_{\mathcal{F}p} A^{-1}_{pp} A_{p\mathcal{F}}$$

global Schur complement matrix

$$S = \sum_{1\le p\le D} \mathcal{R}^{(p)T} S^{(p)} \mathcal{R}^{(p)} \qquad : \text{symmetric positive definite}$$

+ preconditioned CG method

Iterative substructuring method (2/2)

$$\begin{bmatrix} \{A_{pp}\} & \{A_{p\mathcal{F}}\} \\ \{A_{\mathcal{F}p}\} & A_{\mathcal{F}\mathcal{F}} \end{bmatrix} =$$

$$\begin{bmatrix} \{I_p\} & 0 \\ \{A_{\mathcal{F}p}A_{pp}^{-1}\} & I_{\mathcal{F}} \end{bmatrix} \begin{bmatrix} \{A_{pp}\} & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} \{I_p\} & \{A_{pp}^{-1}A_{p\mathcal{F}}\} \\ 0 & I_{\mathcal{F}} \end{bmatrix}$$

A : SPD $\Rightarrow$ S: SPD

by Sylvester's law of inertia.

Neumann-Neumann preconditionder: (1/2)

$$\text{Ker} \begin{bmatrix} A_{pp} & A_{p\mathcal{F}} \\ A_{\mathcal{F}p} & A^{(p)}_{\mathcal{F}\mathcal{F}} \end{bmatrix} = \text{span}[\left\{ \begin{bmatrix} \omega_p^{(i)} \\ \omega^{(i)}_{\mathcal{F},p} \end{bmatrix} \right\}]$$

elasticity problem $\rightarrow$ rigid body movements

$$\text{Ker} S^{(p)} = \text{span}[\omega^{(i)}_{\mathcal{F},p}]$$

$$S = \sum_{1 \leq p \leq D} \mathcal{R}^{(p)T} S^{(p)} \mathcal{R}^{(p)}$$

1-level Neumann-Neumann preconditioner:

[De Roeck-Le Tallec 1991]

$$S^{\dagger} \sim \sum_{1 \leq p \leq D} \mathcal{R}^{(p)T} D^{(p)} S^{(p)\dagger} \mathcal{D}^{(p)} \mathcal{R}^{(p)} .$$

$\mathcal{D}^{(p)}$: partition of unity, $\sum_{1 \leq p \leq D} \mathcal{R}^{(p)^T} \mathcal{D}^{(p)} \mathcal{R}^{(p)} = I_{\mathcal{F}}$.

Neumann-Neumann preconditioner: (2/2)

$$\begin{bmatrix} A_{p\,p} & A_{p\,\mathcal{F}} \\ A_{\mathcal{F}\,p} & A^{(p)}_{\mathcal{F}\,\mathcal{F}} \end{bmatrix} \begin{bmatrix} u_p \\ u_{\mathcal{F},p} \end{bmatrix} = \begin{bmatrix} 0 \\ r_{\mathcal{F},p} \end{bmatrix} \quad \text{on span}[\{ \begin{bmatrix} \omega_p^{(i)} \\ \omega_{\mathcal{F},\,p}^{(i)} \end{bmatrix} \}]^{\perp}$$

$$\begin{bmatrix} A_{p\,p} & A_{p\,\mathcal{F}} \\ 0 & S^{(p)} \end{bmatrix} \begin{bmatrix} u_p \\ u_{\mathcal{F},p} \end{bmatrix} = \begin{bmatrix} 0 \\ r_{\mathcal{F},p} \end{bmatrix} \quad \text{on span}[\{ \begin{bmatrix} \omega_p^{(i)} \\ \omega_{\mathcal{F},\,p}^{(i)} \end{bmatrix} \}]^{\perp}$$

$$S^{(p)} u_{\mathcal{F},p} = r_{\mathcal{F},p} \quad \text{on span}[\{ \left[\omega_{\mathcal{F},\,p}^{(i)} \right] \}]^{\perp}$$

local Schur complement matrix

$\Leftrightarrow$ subproblems with Neumann B.C.

Balancing Neumann-Neumann preconditioner

subspace for coarse grid correction [Mandel 1993]

$$Y := \{w \in \mathbb{R}^{N_{\mathcal{F}}} \, ; \, w = \sum_{1 \le p \le D} R^{(p)T} D^{(p)} v^{(p)} \, , \; v^{(p)} \in \mathrm{Ker} S^{(p)}\} \, .$$

P_Y : orthogonal projection, $\mathbb{R}^{N_{\mathcal{F}}} \rightarrow Y$.

$S_0 = P_Y S P_Y$: Schur complement matrix on the coarse space Y.

balancing Neumann-Neumann preconditioner:

$$Q_{BNN} := (I - S_0^{\dagger}) \left(\sum_{1 \le i \le D} R^{(i)T} D^{(i)} S^{(i)\dagger} D^{(i)} R^{(i)} \right) (I - S_0^{\dagger}) + S_0^{\dagger} \, .$$

subspace for coarse grid correction [Mandel 1993]

$$Y := \{w \in \mathbb{R}^{N_{\mathcal{F}}} \,;\, w = \sum_{1 \le p \le D} R^{(p)T} D^{(p)} v^{(p)} \,,\; v^{(p)} \in \mathrm{Ker} S^{(p)}\} \,.$$

r: residual $\Rightarrow$ $M_{\mathrm{BNN}} r$: preconditioning.

(1) Find $x \in Y$ such that $(S\,x, w) = (r, w) \;\; \forall w \in Y$.

(2) $r^{(p)} := D^{(p)} R^{(p)} (r - S\,x)$ $(\in \mathrm{Im} S^{(p)})$.

(3) Find $u^{(p)} \in \mathrm{Im} S^{(p)}$ such that $S^{(p)} u^{(p)} = r^{(p)}$.

(4) $u := \sum_p R^{(p)T} D^{(p)} u^{(p)}$.

(5) Find $y \in Y$ such that $(S(u+y), w) = (r, w) \;\; \forall w \in Y$.

(6) $M_{\mathrm{BNN}} r = u + y$.

Remark:

(1), (5) : equation of S_0: solvable $\leftarrow$ S : SPD.

(1), (2) $\Leftrightarrow$ $(D^{(p)} R^{(p)} (r - S)\, x \,, v^{(p)}) = 0 \quad \forall v^{(p)} \in \mathrm{ker} S^{(p)}$.

References

- F. Ben Belgacem, Y. Maday. The mortar element method for three-dimensional finite elements, RAIRO Modél. Math. Anal. Numér. 31 (1997) 289–302.
- C. Farhat, F. X. Roux. A method of finite element tearing and interconnecting and its parallel solution algorithm, Int. J. Numer. Methods Eng. 32 (1991) 1205–1227.
- J. Mandel. Balancing domain decomposition, Commun. Numer. Methods Eng. 9 (1993) 233-241.
- `http://www.openmp.org`. OpenMP C and C++ Application Program Interface Version 1.0 – October 1998, Document Number 004-2229-001.
- Y. Saad. Iterative Methods for Sparse Linear Systems 2nd ed., SIAM, 2003.
- A. Toselli, O. B.Widlund. Domain Decomposition Methods: algorithms and theory, Springer, 2004.