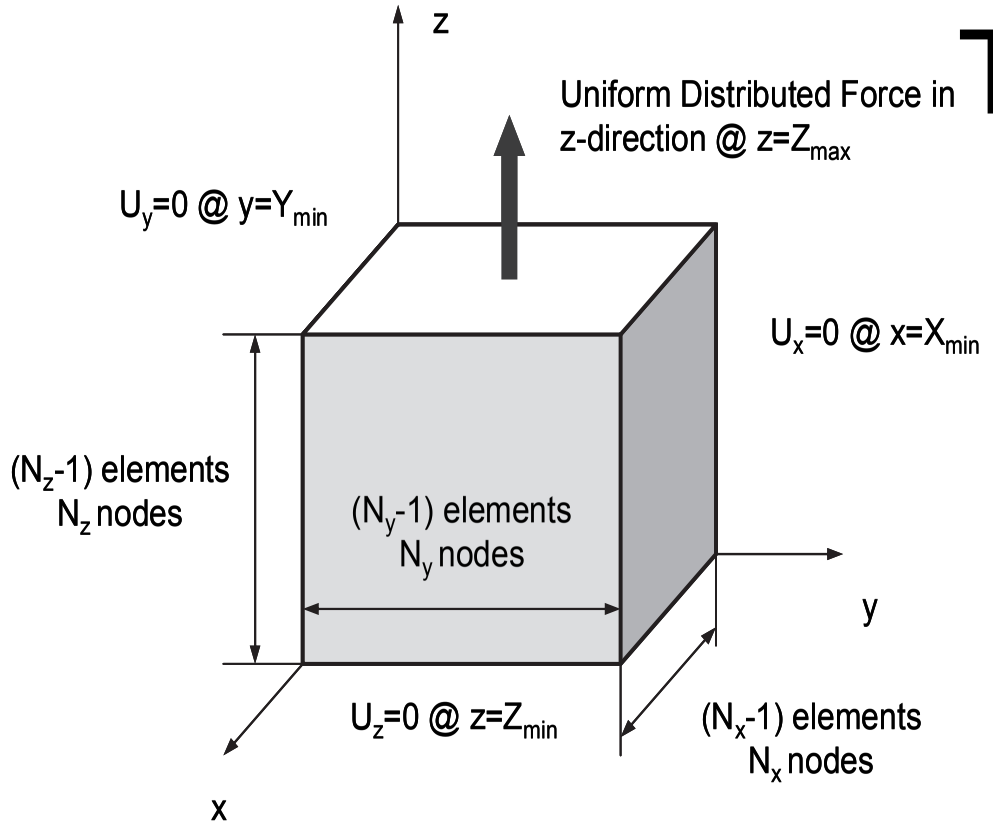
3D Parallel FEM (V) Communication-Computation Overlapping

Kengo Nakajima

Programming for Parallel Computing (616-2057)
Seminar on Advanced Computing (616-4009)

Target Application



- Boundary Conditions
  - Symmetric B.C.
    - $U_X=0@X=0$
    - $U_Y=0@Y=0$
    - $U_Z=0@Z=0$
  - Uniform Distributed Force
    - $F_Z=1@Z=Z_{max}$

- Elastic Material
  - Young's Modulus: E $(=1.00)$, Poisson's Ratio: ν $(=0.30)$
- Rectangular Priem
  - 1x1x1 cubes (hexahedra)
  - NX, NY, NZ nodes in each direction

Overview of the Program

- 3D Static-Linear-Elastic Problem (Solid Mechanics)
  - 3x3 Block Operation
- CG without Preconditioning
- Hybrid
- Parallel distributed meshes are generated in the program automatically.
  - NO need for separate process of mesh generation, and domain partitioning
- Communication-computation overlapping is introduced to SpMV (matrix vector products) of CG
  - Send_Recv: Amount of message is small, most of the time is spent for “latency”

Files ...

Fortran Only

```
>$ cd <$O-TOP>
>$ cp /home/z30088/class_eps/cc_overlap.tar .

>$ cd cc_overlap
>$ cd src0
>$ make

>$ cd ../src0m
>$ make

>$ cd ../run
>$ ls -l sol*
   sol0      No overlapping
   sol0m     Comm.-Comp. Overlapping
```

Execution

Fortran Only

```
>$ cd <$O-TOP>/cc_overlap/run

Please modify the following files:

mesh.inp     Problem Configuration
go00.sh      Batch script file for “sol0”
go0m.sh      Batch script file for “sol0m”
```

“mesh.inp” for Hybrid 16x1

```
(values)    (variables)   (descriptions)

200 200 300 npx,npy,npz    Total number of nodes in X-, Y-,  and Z-
                           direction (Nx, Ny, Nz in the prev. page
  2   2   3 ndx,ndy,ndz    Partition # in each direction (X,Y,Z)
 16   1     PEsmpTOT,(unused)  Thread # on each MPI proc., unused
200          ITERmax            Number of iterations for CG method
```

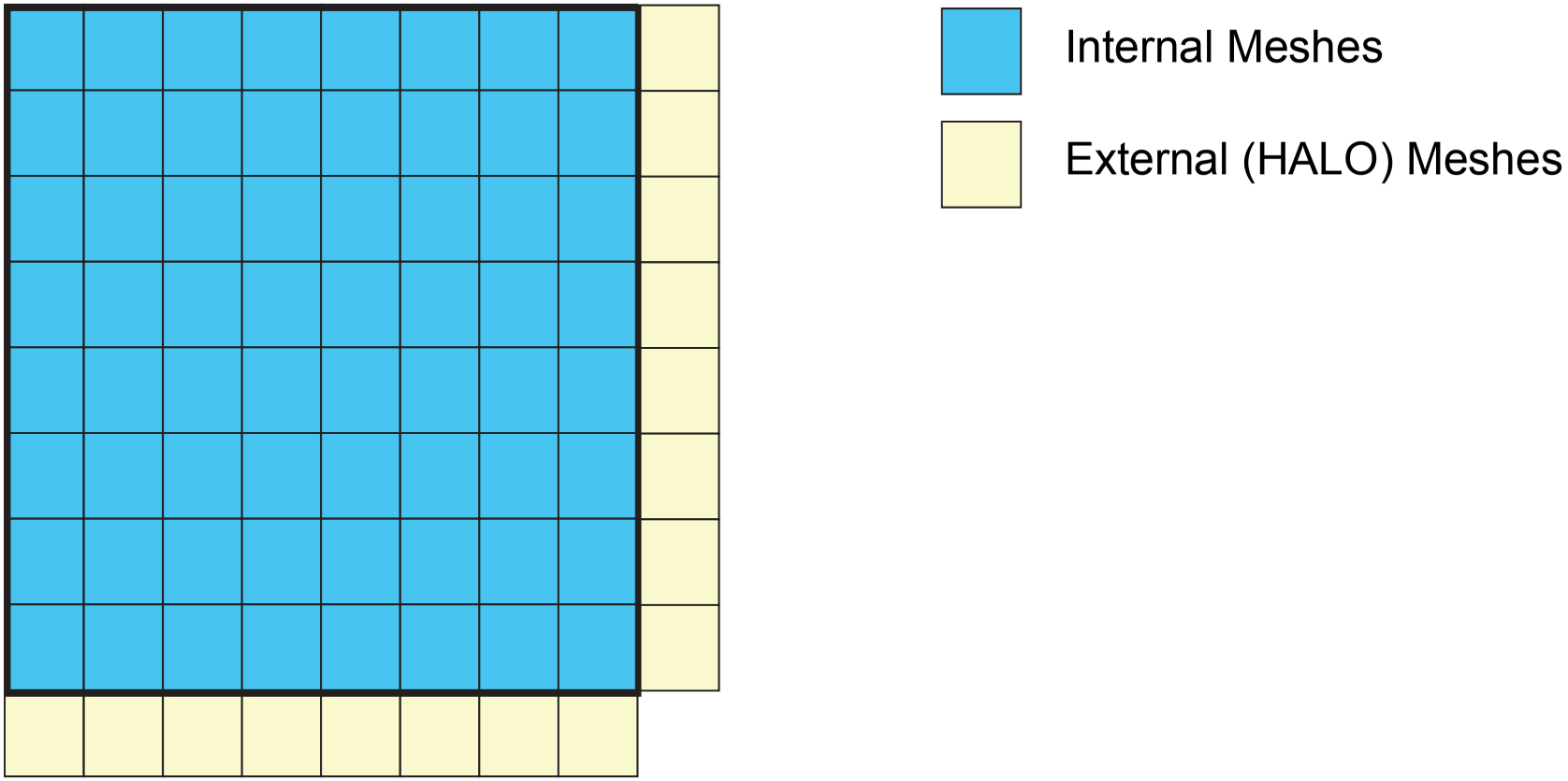
- Each of “npx,npy,npz” must be “divisible（割り切れる）” by each of “ndx,ndy,ndz”
- MPI process # = ndx × ndy × ndz
  - Example: 12 nodes, 192 cores, 12 processes, 16 threads
- “ITERmax” can be small, if you want evaluate the perfomance (it takes a long time until convergence).

Example of “go00.sh”

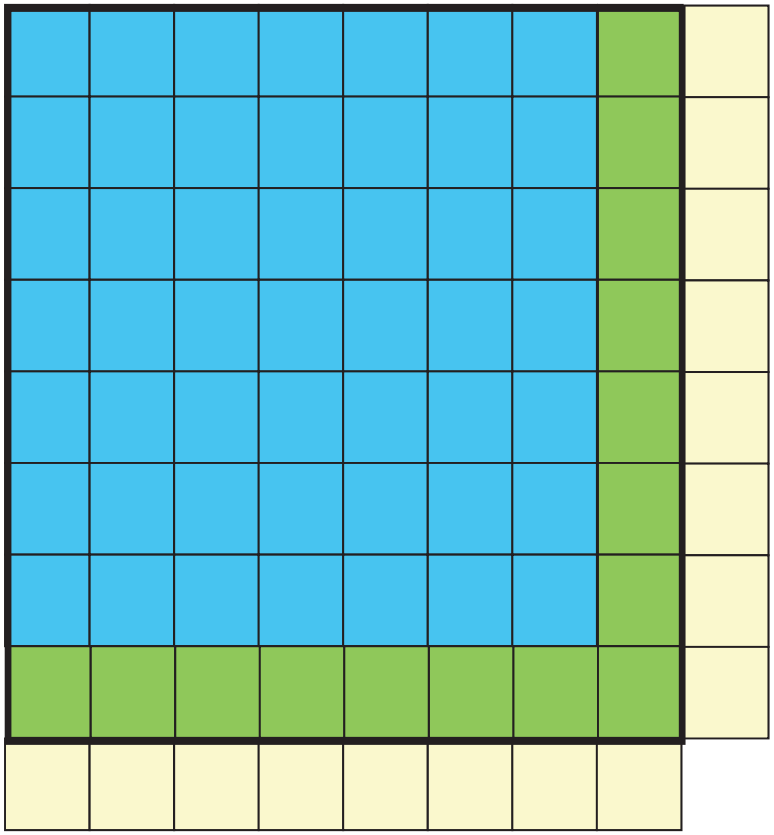
export OMP_NUM_THREADS=PEsmpTOT

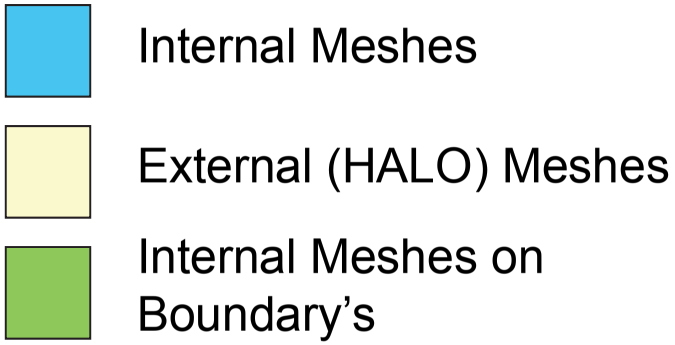
```
#!/bin/sh
#PJM -L "node=12"
#PJM -L "elapse=00:05:00"
#PJM -j
#PJM -L “rscgrp=lecture2“
#PJM -g “gt82“
#PJM -o "t00-012-128-128-128-03.lst"
#PJM --mpi "proc=12"

export OMP_NUM_THREADS=16
mpiexec ./sol0
rm wk.*
```

Comm.-Comp. Overlapping



Comm.-Comp. Overlapping





Mat-Vec operations (SpMV)

- Renumbering: ■ ⇒ ■
- Communications of info. on external meshes
- Computation of ■ BEFORE completion of comm. (comm.-comp. overlapping)
- Synchronization of communications
- Computation of ■

q=Ap in src0

```
      call SOLVER_SEND_RECV_3                                          &
     &    ( N, NP, NEIBPETOT, NEIBPE, STACK_IMPORT, NOD_IMPORT,        &
     &      STACK_EXPORT, NOD_EXPORT, WS, WR, WW(1,P) , SOLVER_COMM,   &
     &      my_rank)

!$omp parallel do private (j,k,i,X1,X2,X3,WVAL1,WVAL2,WVAL3)
      do j= 1, N
           X1= WW(3*j-2,P)
           X2= WW(3*j-1,P)
           X3= WW(3*j  ,P)
        WVAL1= D(9*j-8)*X1 + D(9*j-7)*X2 + D(9*j-6)*X3
        WVAL2= D(9*j-5)*X1 + D(9*j-4)*X2 + D(9*j-3)*X3
        WVAL3= D(9*j-2)*X1 + D(9*j-1)*X2 + D(9*j  )*X3
        do k= INL(j-1)+1, INL(j)
           i= IAL(k)
          X1= WW(3*i-2,P)
          X2= WW(3*i-1,P)
          X3= WW(3*i  ,P)
          WVAL1= WVAL1 + AL(9*k-8)*X1 + AL(9*k-7)*X2 + AL(9*k-6)*X3
          WVAL2= WVAL2 + AL(9*k-5)*X1 + AL(9*k-4)*X2 + AL(9*k-3)*X3
          WVAL3= WVAL3 + AL(9*k-2)*X1 + AL(9*k-1)*X2 + AL(9*k  )*X3
        enddo
        do k= INU(j-1)+1, INU(j)
           i= IAU(k)
          X1= WW(3*i-2,P)
          X2= WW(3*i-1,P)
          X3= WW(3*i  ,P)
          WVAL1= WVAL1 + AU(9*k-8)*X1 + AU(9*k-7)*X2 + AU(9*k-6)*X3
          WVAL2= WVAL2 + AU(9*k-5)*X1 + AU(9*k-4)*X2 + AU(9*k-3)*X3
          WVAL3= WVAL3 + AU(9*k-2)*X1 + AU(9*k-1)*X2 + AU(9*k  )*X3
        enddo

        WW(3*j-2,Q)= WVAL1
        WW(3*j-1,Q)= WVAL2
        WW(3*j  ,Q)= WVAL3
      enddo
```

q=Ap in src0m (1/3)

```
      do neib= 1, NEIBPETOT
        istart= STACK_EXPORT(neib-1)
        inum  = STACK_EXPORT(neib  ) - istart
!$omp parallel do private (ii)
        do k= istart+1, istart+inum
              ii   = 3*NOD_EXPORT(k)
           WS(3*k-2)= WW(ii-2,P)
           WS(3*k-1)= WW(ii-1,P)
           WS(3*k  )= WW(ii  ,P)
        enddo
        call MPI_ISEND (WS(3*istart+1), 3*inum,MPI_DOUBLE_PRECISION,     &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD, req1(neib),     &
     &                  ierr)
      enddo

      do neib= 1, NEIBPETOT
        istart= STACK_IMPORT(neib-1)
        inum  = STACK_IMPORT(neib  ) - istart
        call MPI_IRECV (WW(3*(istart+N)+1,P), 3*inum,                    &
     &                  MPI_DOUBLE_PRECISION,                            &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD,                 &
     &                  req1(neib+NEIBPETOT), ierr)
      enddo

!C
!C-- Pure Inner Nodes

!$omp parallel do private (j,k,i,X1,X2,X3,WVAL1,WVAL2,WVAL3)
      do j= 1, Ninn
           X1= WW(3*j-2,P)
           X2= WW(3*j-1,P)
           X3= WW(3*j  ,P)
        WVAL1= D(9*j-8)*X1 + D(9*j-7)*X2 + D(9*j-6)*X3
        WVAL2= D(9*j-5)*X1 + D(9*j-4)*X2 + D(9*j-3)*X3
        WVAL3= D(9*j-2)*X1 + D(9*j-1)*X2 + D(9*j  )*X3
```

q=Ap in src0m (2/3)

```
!C
!C-- Pure Inner Nodes

!$omp parallel do private (j,k,i,X1,X2,X3,WVAL1,WVAL2,WVAL3)
      do j= 1, Ninn
           X1= WW(3*j-2,P)
           X2= WW(3*j-1,P)
           X3= WW(3*j  ,P)
        WVAL1= D(9*j-8)*X1 + D(9*j-7)*X2 + D(9*j-6)*X3
        WVAL2= D(9*j-5)*X1 + D(9*j-4)*X2 + D(9*j-3)*X3
        WVAL3= D(9*j-2)*X1 + D(9*j-1)*X2 + D(9*j  )*X3
        do k= INL(j-1)+1, INL(j)
           i= IAL(k)
          X1= WW(3*i-2,P)
          X2= WW(3*i-1,P)
          X3= WW(3*i  ,P)
          WVAL1= WVAL1 + AL(9*k-8)*X1 + AL(9*k-7)*X2 + AL(9*k-6)*X3
          WVAL2= WVAL2 + AL(9*k-5)*X1 + AL(9*k-4)*X2 + AL(9*k-3)*X3
          WVAL3= WVAL3 + AL(9*k-2)*X1 + AL(9*k-1)*X2 + AL(9*k  )*X3
        enddo
        do k= INU(j-1)+1, INU(j)
           i= IAU(k)
          X1= WW(3*i-2,P)
          X2= WW(3*i-1,P)
          X3= WW(3*i  ,P)
          WVAL1= WVAL1 + AU(9*k-8)*X1 + AU(9*k-7)*X2 + AU(9*k-6)*X3
          WVAL2= WVAL2 + AU(9*k-5)*X1 + AU(9*k-4)*X2 + AU(9*k-3)*X3
          WVAL3= WVAL3 + AU(9*k-2)*X1 + AU(9*k-1)*X2 + AU(9*k  )*X3
        enddo

        WW(3*j-2,Q)= WVAL1
        WW(3*j-1,Q)= WVAL2
        WW(3*j  ,Q)= WVAL3
      enddo

      call MPI_WAITALL (2*NEIBPETOT, req1, sta1, ierr)        ここで同期をとる
```

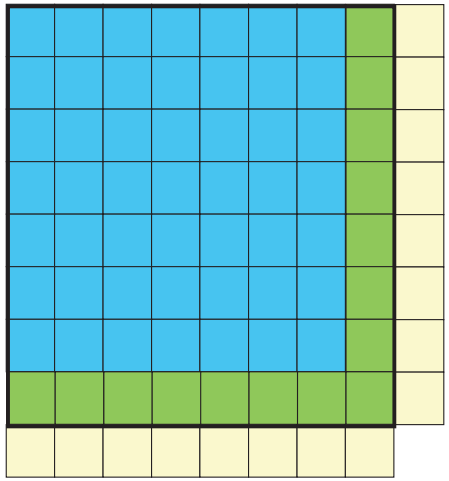
q=Ap in src0m (3/3)

```
!C
!C-- Boundary Nodes

!$omp parallel do private (j,k,i,X1,X2,X3,WVAL1,WVAL2,WVAL3)
      do j= Ninn+1, N
           X1= WW(3*j-2,P)
           X2= WW(3*j-1,P)
           X3= WW(3*j  ,P)
        WVAL1= D(9*j-8)*X1 + D(9*j-7)*X2 + D(9*j-6)*X3
        WVAL2= D(9*j-5)*X1 + D(9*j-4)*X2 + D(9*j-3)*X3
        WVAL3= D(9*j-2)*X1 + D(9*j-1)*X2 + D(9*j  )*X3
        do k= INL(j-1)+1, INL(j)
           i= IAL(k)
          X1= WW(3*i-2,P)
          X2= WW(3*i-1,P)
          X3= WW(3*i  ,P)
          WVAL1= WVAL1 + AL(9*k-8)*X1 + AL(9*k-7)*X2 + AL(9*k-6)*X3
          WVAL2= WVAL2 + AL(9*k-5)*X1 + AL(9*k-4)*X2 + AL(9*k-3)*X3
          WVAL3= WVAL3 + AL(9*k-2)*X1 + AL(9*k-1)*X2 + AL(9*k  )*X3
        enddo
        do k= INU(j-1)+1, INU(j)
           i= IAU(k)
          X1= WW(3*i-2,P)
          X2= WW(3*i-1,P)
          X3= WW(3*i  ,P)
          WVAL1= WVAL1 + AU(9*k-8)*X1 + AU(9*k-7)*X2 + AU(9*k-6)*X3
          WVAL2= WVAL2 + AU(9*k-5)*X1 + AU(9*k-4)*X2 + AU(9*k-3)*X3
          WVAL3= WVAL3 + AU(9*k-2)*X1 + AU(9*k-1)*X2 + AU(9*k  )*X3
        enddo

        WW(3*j-2,Q)= WVAL1
        WW(3*j-1,Q)= WVAL2
        WW(3*j  ,Q)= WVAL3
      enddo
```

Comm.-Comp. Overlapping

With Reordering (current)

```
      call MPI_Isend
      call MPI_Irecv

      do i= 1, Ninn
        (calculations)
      enddo

      call MPI_Waitall

      do i= Ninn+1, Nall
        (calculationas)
      enddo
```



Without Reordering

```
      call MPI_Isend
      call MPI_Irecv

      do i= 1, Nall
        if (INNflag(i).eq.1) then
          (calculations)
        endif
      enddo

      call MPI_Waitall

      do i= 1, Nall
        if (INNflag(i).eq.0) then
          (calculations)
        endif
      enddo
```

Results: Elapsed time for a single q=Ap operation in CG (sec.)

for 12 nodes (processes)

Sufficient computation time for overlapping is needed
Amount of computation for each node must be large !

	(npx,npy,npz)(ndx,ndy,ndz)	
	(256,256,384) (2,2,3)	(200,200,300) (2,2,3)
sol0	1.11×10^{-1}	4.13×10^{-2}
sol0m	9.08×10^{-2}	4.12×10^{-2}

Summary

- Try various cases
  - Problem size
  - Number of threads per MPI process
  - Effect of reordering (change of the program)
- Comm.-Comp. Overlapping
  - Significant effects are not expected for SpMV processes
  - Explicit time marching method with large computational amount in each loop
  - Complicated parallel preconditioning
    - area of on-going research