Introduction to Parallel Programming for Multicore/Manycore Clusters

Part V: Parallel Version by OpenMP

Kengo Nakajima
Information Technology Center
The University of Tokyo

Parallel Version: OpenMP

- OpenMP version of L2-sol
  - Number of threads= "PEsmpTOT"
    - can be controlled in the program

- Fundamental Idea
  - Meshes in a same color/level are independent, therefore parallel/concurrent processing is possible for these meshes.

4-Colors, 4-Threads
Initial Mesh

57	58	59	60	61	62	63	64
49	50	51	52	53	54	55	56
41	42	43	44	45	46	47	48
33	34	35	36	37	38	39	40
25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24
9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8

4-Colors, 4-Threads
Initial Mesh

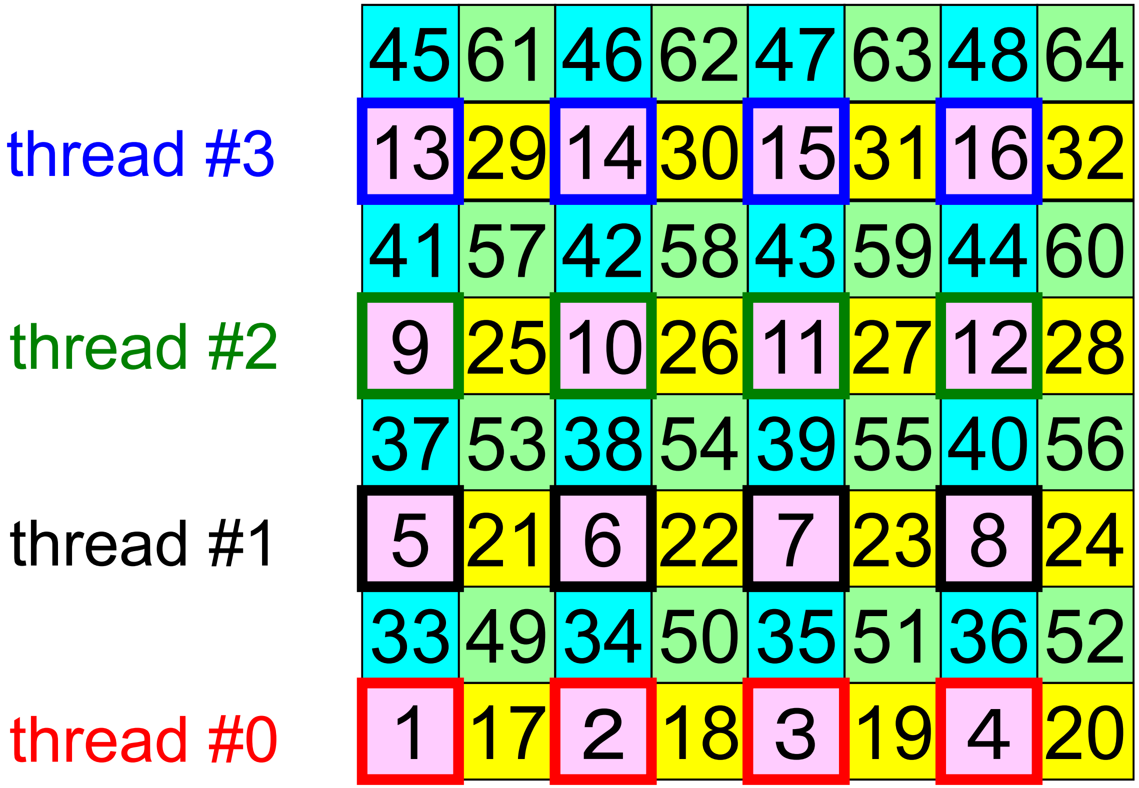
57	58	59	60	61	62	63	64
49	50	51	52	53	54	55	56
41	42	43	44	45	46	47	48
33	34	35	36	37	38	39	40
25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24
9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8

4-Colors, 4-Threads
Renumbering according to Color ID

45	61	46	62	47	63	48	64
13	29	14	30	15	31	16	32
41	57	42	58	43	59	44	60
9	25	10	26	11	27	12	28
37	53	38	54	39	55	40	56
5	21	6	22	7	23	8	24
33	49	34	50	35	51	36	52
1	17	2	18	3	19	4	20

4-Colors, 4-Threads

Meshes in a same color/level are independent, therefore parallel/concurrent processing is possible for these meshes, renumbered meshes are assigned to threads

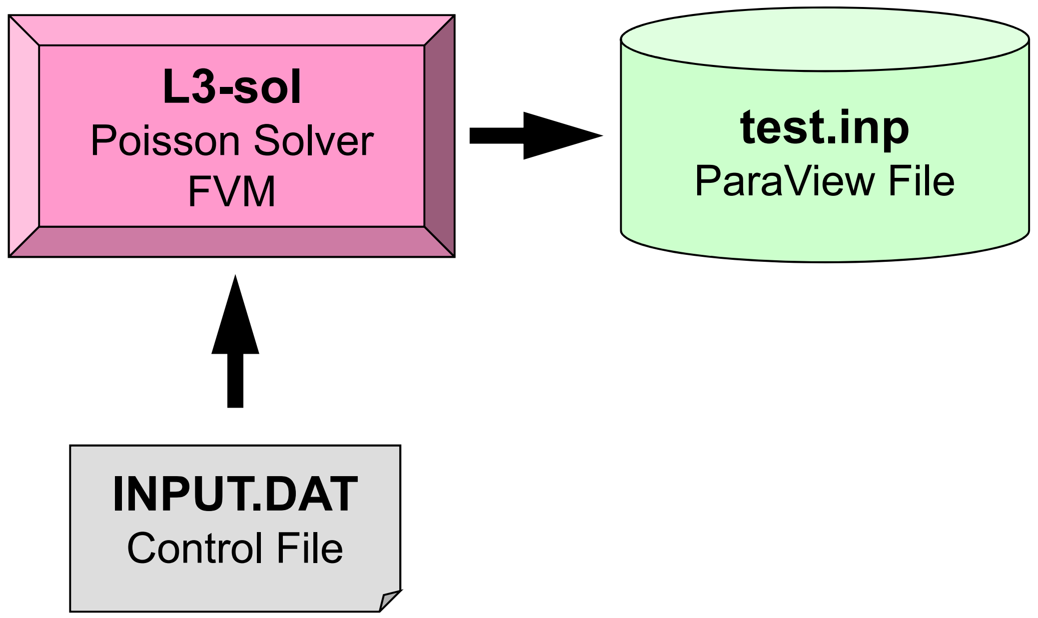


Files on FX10

```
>$ cd
>$ cp /home/z30088/omp/omp-f5.tar .
>$ tar xvf omp-f5.tar

>$ cd multicore/L3                    <$O-L3>

>$ ls
reorder0 run src src0
```

```
% cd <$O-L3>
% ls
    run   src  src0  reorder0
% cd src
% make
% cd ../run
% ls L3-sol
    L3-sol
% <modify "<$O-L3>/run/INPUT.DAT">
% <modify "<$O-L3>/run/go1">
% pjsub go1.sh
```

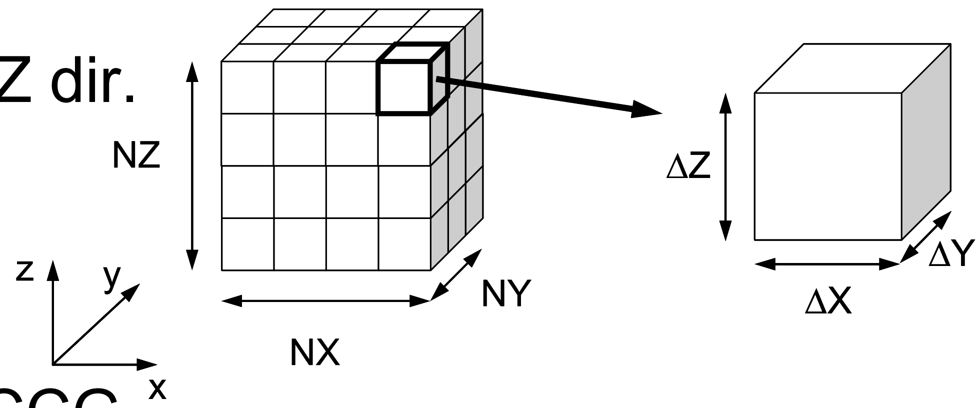
Running the Program



Control Data: INPUT.DAT

```
100 100 100                                 NX/NY/NZ
1.00e-00 1.00e-00 1.00e-00                  DX/DY/DZ
1.0e-08                                     EPSICCG
16                                          PEsmpTOT
100                                         NCOLORtot
```

- **NX,NY,NZ**
  - Number of meshes in X/Y/Z dir.
- **DX,DY,DZ**
  - Size of meshes
- **EPSICCG**
  - Convergence Criteria for ICCG
- **PEsmpTOT**
  - Thread Number **(= OMP_NUM_THREADS)**
- **NCOLORtot**
  - Reordering Method + Initial Number of Colors/Levels
  - ≧2: MC, =0: CM, =-1: RCM, -2≧: CMRCM



go1.sh

```
#!/bin/sh
#PJM -L "node=1"
#PJM -L "elapse=00:10:00"
#PJM -L "rscgrp=lecture9"
#PJM -g "gt99"
#PJM -j
#PJM -o “test.lst"

export OMP_NUM_THREADS=16      =PEsmpTOT
./L3-sol
```

- Applying OpenMP to L2-sol
- Examples
- Optimization + Exercise

Applying OpenMP to “L2-sol”

- on ICCG solver

- Dot Products, DAXPY, Mat-Vec
  - NO data dependency: Just insert directives
- Preconditioning (IC Factorization, Forward/Backward Substitution)
  - NO data dependency in same color: Parallel processing is possible for meshes in same color

Just inserting directives works fine, but ... (1/2) (Mat-Vec)

```
!$omp parallel do private(i,VAL,k)
        do i = 1, N
          VAL= D(i)*W(i,P)
          do k= indexL(i-1)+1, indexL(i)
            VAL= VAL + AL(k)*W(itemL(k),P)
          enddo
          do k= indexU(i-1)+1, indexU(i)
            VAL= VAL + AU(k)*W(itemU(k),P)
          enddo
          W(i,Q)= VAL
        enddo
!$omp end parallel do
```

- Thread number cannot be handled in the program
- But, this may work better on GPU and manycore's

Just inserting directives works fine, but ... (2/2) (Forward Substitution)

```
      do icol= 1, NCOLORtot
!$omp parallel do private (i, VAL, k)
        do i= COLORindex(icol-1)+1, COLORindex(icol)
          VAL= D(i)
          do k= indexL(i-1)+1, indexL(i)
            VAL= VAL - (AL(k)**2) * DD(itemL(k))
          enddo
          DD(i)= 1.d0/VAL
        enddo
!$omp end parallel do
      enddo
```

- Thread number cannot be handled in the program
- But, this may work better on GPU and manycore's

Parallelize ICCG Method by OpenMP

- Dot Product: OK
- DAXPY: OK
- Matrix-Vector Multiply: OK
- Preconditioning

Main Program

```
      program MAIN

      use STRUCT
      use PCG
      use solver_ICCG_mc

      implicit REAL*8 (A-H,O-Z)
      real(kind=8), dimension(:), allocatable :: WK

      call INPUT
      call POINTER_INIT
      call BOUNDARY_CELL
      call CELL_METRICS
      call POI_GEN

      PHI=  0.d0

      call solve_ICCG_mc                                              &
     &      ( ICELTOT, NPL, NPU, indexL, itemL, indexU, itemU, D,      &
     &        BFORCE,  PHI, AL, AU, NCOLORtot, PEsmpTOT,               &
     &        SMPindex, SMPindexG, EPSICCG, ITR, IER)

      allocate (WK(ICELTOT))
      do ic0= 1, ICELTOT
        icel= NEWtoOLD(ic0)
        WK(icel)= PHI(ic0)
      enddo

      do icel= 1, ICELTOT
        PHI(icel)= WK(icel)
      enddo

      call OUTUCD

      stop
      end
```

module STRUCT

```
      module STRUCT

      use omp_lib
      include 'precision.inc'

!C
!C-- METRICs & FLUX
      integer (kind=kint) :: ICELTOT, ICELTOTp, N
      integer (kind=kint) :: NX, NY, NZ, NXP1, NYP1, NZP1, IBNODTOT
      integer (kind=kint) :: NXc, NYc, NZc

      real (kind=kreal) ::                                              &
     &      DX, DY, DZ, XAREA, YAREA, ZAREA, RDX, RDY, RDZ,             &
     &      RDX2, RDY2, RDZ2, R2DX, R2DY, R2DZ

      real (kind=kreal), dimension(:), allocatable ::                   &
     &     VOLCEL, VOLNOD, RVC, RVN

      integer (kind=kint), dimension(:,:), allocatable ::               &
     &         XYZ, NEIBcell

!C
!C-- BOUNDARYs
      integer (kind=kint) :: ZmaxCELtot
      integer (kind=kint), dimension(:), allocatable :: BC_INDEX, BC_NOD
      integer (kind=kint), dimension(:), allocatable :: ZmaxCEL

!C
!C-- WORK
      integer (kind=kint), dimension(:,:), allocatable :: IWKX
      real(kind=kreal),    dimension(:,:), allocatable :: FCV

      integer (kind=kint) :: PEsmpTOT

      end module STRUCT
```

ICELTOT:
Number of meshes (NX x NY x NZ)

N:
Number of modes

NX,NY,NZ:
Number of meshes in x/y/z directions

NXP1,NYP1,NZP1:
Number of nodes in x/y/z directions

IBNODTOT:
= NXP1 x NYP1

XYZ(ICELTOT,3):
Location of meshes

NEIBcell(ICELTOT,6):
Neighboring meshes

PEsmpTOT:
Number of threads

module PCG

```
module PCG

integer, parameter :: N2= 256
integer :: NUmax, NLmax, NCOLORtot, NCOLORk, NU, NL
integer :: NPL, NPU
integer :: METHOD, ORDER_METHOD

real(kind=8) :: EPSICCG

real(kind=8), dimension(:), allocatable :: D, PHI, BFORCE
real(kind=8), dimension(:), allocatable :: AL, AU

integer, dimension(:), allocatable :: INL, INU, COLORindex
integer, dimension(:), allocatable :: SMPindex, SMPindexG
integer, dimension(:), allocatable :: OLDtoNEW, NEWtoOLD

integer, dimension(:,:), allocatable :: IAL, IAU

integer, dimension(:), allocatable :: indexL, itemL
integer, dimension(:), allocatable :: indexU, itemU
end module PCG
```

`NCOLORtot`	Total number of colors/levels
`COLORindex`	Index of number of meshes in each color/level
`(0:NCOLORtot)`	`(COLORindex(icol)- COLORindex(icol-1))`

`SMPindex (0:NCOLORtot*PEsmpTOT)`
`SMPindexG(0:PEsmpTOT)`

`OLDtoNEW, NEWtoOLD` Reference table before/after renumbering

Variables/Arrays for Matrix (1/2)

Name	Type	Content
D(N)	**R**	Diagonal components of the matrix (N= ICELTOT)
BFORCE(N)	**R**	RHS vector
PHI(N)	**R**	Unknown vector
indexL(0:N), indexU(0:N)	**I**	# of L/U non-zero off-diag. comp. (CRS)
NPL, NPU	**I**	Total # of L/U non-zero off-diag. comp. (CRS)
itemL(NPL), itemU(NPU)	**I**	Column ID of L/U non-zero off-diag. comp. (CRS)
AL(NPL), AU(NPU)	**R**	L/U non-zero off-diag. comp. (CRS)

Name	Type	Content
NL,NU	**I**	MAX. # of L/U non-zero off-diag. comp. for each mesh (=6)
INL(N), INU(N)	**I**	# of L/U non-zero off-diag. comp.
IAL(NL,N), IAU(NU,N)	**I**	Column ID of L/U non-zero off-diag. comp.

Variables/Arrays for Matrix (2/2)

Name	Type	Content
`NCOLORtot`	`I`	**Input:** reordering method + initial number of colors/levels ≧2: MC, =0: CM, =-1: RCM, -2≧: CMRCM **Output:** Final number of colors/levels
`COLORindex (0:NCOLORtot)`	`I`	Number of meshes at each color/level 1D compressed array Meshes in `icol`th color/level are stored in this array from `COLORindex(icol-1)+1` to `COLORindex(icol)`
`NEWtoOLD(N)`	`I`	Reference array from New to Old numbering
`OLDtoNEW(N)`	`I`	Reference array from Old to New numbering
`PEsmpTOT`	`I`	Number of Threads
`SMPindex (0:NCOLORtot*PEsmpTOT)`	`I`	Array for OpenMP Operations (for Loops with Data Dependency)
`SMPindexG(0:PEsmpTOT)`	`I`	Array for OpenMP Operations (for Loops without Data Dependency)

Main Program

```
      program MAIN

      use STRUCT
      use PCG
      use solver_ICCG_mc

      implicit REAL*8 (A-H,O-Z)
      real(kind=8), dimension(:), allocatable :: WK

      call INPUT
      call POINTER_INIT
      call BOUNDARY_CELL
      call CELL_METRICS
      call POI_GEN

      PHI=  0.d0

      call solve_ICCG_mc                                                &
     &    ( ICELTOT, NPL, NPU, indexL, itemL, indexU, itemU, D,        &
     &      BFORCE,  PHI, AL, AU, NCOLORtot, PEsmpTOT,                 &
     &      SMPindex, SMPindexG, EPSICCG, ITR, IER)
```

input: reading INPUT.DAT

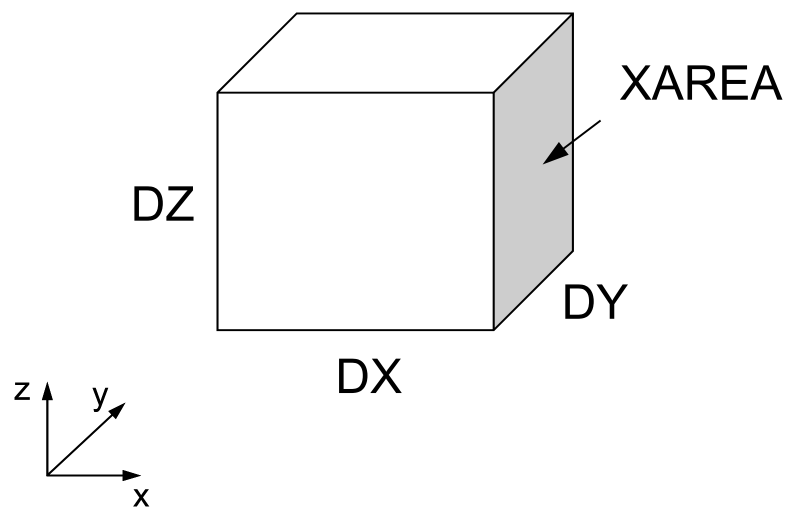
```
!C
!C***
!C*** INPUT
!C***
!C
!C    INPUT CONTROL DATA
!C
      subroutine INPUT
      use STRUCT
      use PCG
      implicit REAL*8 (A-H,O-Z)
      character*80 CNTFIL
!C
!C-- CNTL. file
      open  (11, file='INPUT.DAT', status='unknown')
        read (11,*) NX, NY, NZ
        read (11,*) DX, DY, DZ
        read (11,*) EPSICCG
        read (11,*) PEsmpTOT
        read (11,*) NCOLORtot
      close (11)
!C===
      return
      end
```

- **PEsmpTOT**
  - Thread Number
- **NCOLORtot**
  - Reordering Method + Initial Number of Colors/Levels
  - ≧2: MC
  - =0: CM
  - =-1: RCM
  - -2≧: CMRCM

```
100 100 100                        NX/NY/NZ
1.00e-02 5.00e-02 1.00e-02         DX/DY/DZ
1.00e-08                           EPSICCG
16                                 PEsmpTOT
100                                NCOLORtot
```

cell_metrics

```
!C
!C***
!C*** CELL_METRICS
!C***
!C
      subroutine CELL_METRICS
      use STRUCT
      use PCG
      implicit REAL*8 (A-H,O-Z)

!C
!C-- ALLOCATE
      allocate (VOLCEL(ICELTOT))
      allocate (   RVC(ICELTOT))
!C
!C-- VOLUME, AREA, PROJECTION etc.
      XAREA= DY * DZ
      YAREA= DX * DZ
      ZAREA= DX * DY

      RDX= 1.d0 / DX
      RDY= 1.d0 / DY
      RDZ= 1.d0 / DZ

      RDX2= 1.d0 / (DX**2)
      RDY2= 1.d0 / (DY**2)
      RDZ2= 1.d0 / (DZ**2)

      R2DX= 1.d0 / (0.50d0*DX)
      R2DY= 1.d0 / (0.50d0*DY)
      R2DZ= 1.d0 / (0.50d0*DZ)

       V0= DX * DY * DZ
      RV0= 1.d0/V0

      VOLCEL=  V0
      RVC   = RV0

      return
      end
```



Main Program

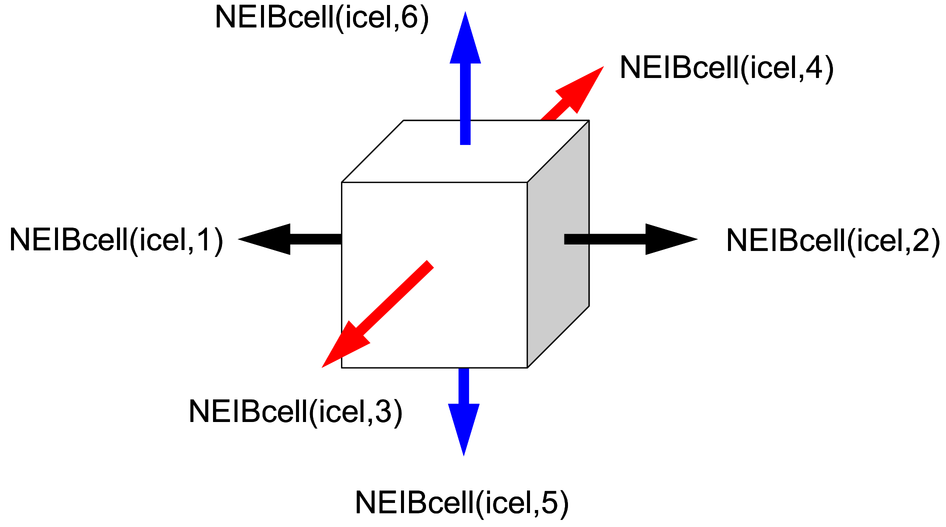
```
      program MAIN

      use STRUCT
      use PCG
      use solver_ICCG_mc

      implicit REAL*8 (A-H,O-Z)
      real(kind=8), dimension(:), allocatable :: WK

      call INPUT
      call POINTER_INIT
      call BOUNDARY_CELL
      call CELL_METRICS
      call POI_GEN

      PHI=  0.d0

        call solve_ICCG_mc                                            &
     &      ( ICELTOT, NPL, NPU, indexL, itemL, indexU, itemU, D,     &
     &        BFORCE,  PHI, AL, AU, NCOLORtot, PEsmpTOT,              &
     &        SMPindex, SMPindexG, EPSICCG, ITR, IER)
```

poi_gen (1/9)

```
      subroutine POI_GEN

      use STRUCT
      use PCG

      implicit REAL*8 (A-H,O-Z)

!C
!C-- INIT.
      nn = ICELTOT
      nnp= ICELTOTp

      NU= 6
      NL= 6

      allocate (BFORCE(nn), D(nn), PHI(nn))
      allocate (INL(nn), INU(nn), IAL(NL,nn), IAU(NU,nn))

      PHI   = 0.d0
        D   = 0.d0
      BFORCE= 0.d0

      INL= 0
      INU= 0
      IAL= 0
      IAU= 0
```

```
!C
!C +--------------+
!C | CONNECTIVITY |
!C +--------------+
!C===
      do icel= 1, ICELTOT
        icN1= NEIBcell(icel,1)
        icN2= NEIBcell(icel,2)
        icN3= NEIBcell(icel,3)
        icN4= NEIBcell(icel,4)
        icN5= NEIBcell(icel,5)
        icN6= NEIBcell(icel,6)

        if (icN5.ne.0.and.icN5.le.ICELTOT) then
          icou= INL(icel) + 1
          IAL(icou,icel)= icN5
          INL(     icel)= icou
        endif

        if (icN3.ne.0.and.icN3.le.ICELTOT) then
          icou= INL(icel) + 1
          IAL(icou,icel)= icN3
          INL(     icel)= icou
        endif

        if (icN1.ne.0.and.icN1.le.ICELTOT) then
          icou= INL(icel) + 1
          IAL(icou,icel)= icN1
          INL(     icel)= icou
        endif

        if (icN2.ne.0.and.icN2.le.ICELTOT) then
          icou= INU(icel) + 1
          IAU(icou,icel)= icN2
          INU(     icel)= icou
        endif

        if (icN4.ne.0.and.icN4.le.ICELTOT) then
          icou= INU(icel) + 1
          IAU(icou,icel)= icN4
          INU(     icel)= icou
        endif

        if (icN6.ne.0.and.icN6.le.ICELTOT) then
          icou= INU(icel) + 1
          IAU(icou,icel)= icN6
          INU(     icel)= icou
        endif
      enddo
!C===
```

poi_gen (2/9)

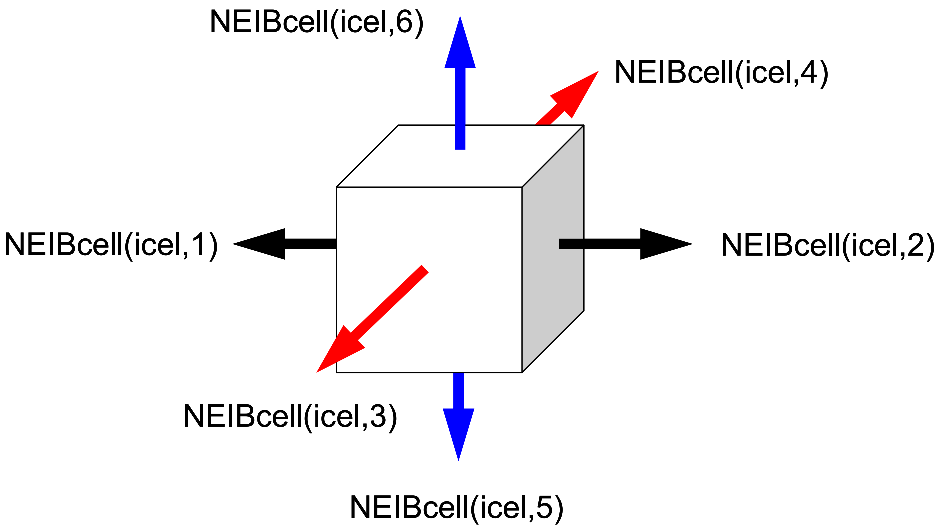


Lower Triangular Part

NEIBcell(icel,5)= icel – NX*NY
NEIBcell(icel,3)= icel – NX
NEIBcell(icel,1)= icel – 1

poi_gen (2/9)

```
!C
!C +--------------+
!C | CONNECTIVITY |
!C +--------------+
!C===
      do icel= 1, ICELTOT
        icN1= NEIBcell(icel,1)
        icN2= NEIBcell(icel,2)
        icN3= NEIBcell(icel,3)
        icN4= NEIBcell(icel,4)
        icN5= NEIBcell(icel,5)
        icN6= NEIBcell(icel,6)

        if (icN5.ne.0.and.icN5.le.ICELTOT) then
          icou= INL(icel) + 1
          IAL(icou,icel)= icN5
          INL(     icel)= icou
        endif

        if (icN3.ne.0.and.icN3.le.ICELTOT) then
          icou= INL(icel) + 1
          IAL(icou,icel)= icN3
          INL(     icel)= icou
        endif

        if (icN1.ne.0.and.icN1.le.ICELTOT) then
          icou= INL(icel) + 1
          IAL(icou,icel)= icN1
          INL(     icel)= icou
        endif

        if (icN2.ne.0.and.icN2.le.ICELTOT) then
          icou= INU(icel) + 1
          IAU(icou,icel)= icN2
          INU(     icel)= icou
        endif

        if (icN4.ne.0.and.icN4.le.ICELTOT) then
          icou= INU(icel) + 1
          IAU(icou,icel)= icN4
          INU(     icel)= icou
        endif

        if (icN6.ne.0.and.icN6.le.ICELTOT) then
          icou= INU(icel) + 1
          IAU(icou,icel)= icN6
          INU(     icel)= icou
        endif
      enddo
!C===
```



Upper Triangular Part

NEIBcell(icel,2)= icel + 1
NEIBcell(icel,4)= icel + NX
NEIBcell(icel,6)= icel + NX*NY

poi_gen (3/9)

```
!C
!C +----------------+
!C | MULTICOLORING |
!C +----------------+
!C===
      allocate (OLDtoNEW(ICELTOT), NEWtoOLD(ICELTOT))
      allocate (COLORindex(0:ICELTOT))

111    continue
        write (*,'(//a,i8,a)') 'You have', ICELTOT, ' elements.'
        write (*,'(  a     )') 'How many colors do you need ?'
        write (*,'(  a     )') '  #COLOR must be more than 2 and'
        write (*,'(  a,i8  )') '  #COLOR must not be more than', ICELTOT
        write (*,'(  a     )') '   CM if #COLOR .eq. 0'
        write (*,'(  a     )') '  RCM if #COLOR .eq.-1'
        write (*,'(  a     )') 'CMRCM if #COLOR .le.-2'
        write (*, '(  a     )' )  '=>'

      if (NCOLORtot.gt.0) then
        call MC  (ICELTOT, NL, NU, INL, IAL, INU, IAU,                &
     &              NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif

      if (NCOLORtot.eq.0) then
        call  CM (ICELTOT, NL, NU, INL, IAL, INU, IAU,                &
     &              NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif

      if (NCOLORtot.eq.-1) then
        call RCM (ICELTOT, NL, NU, INL, IAL, INU, IAU,                &
     &              NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif

      if (NCOLORtot.lt.-1) then
        call CMRCM (ICELTOT, NL, NU, INL, IAL, INU, IAU,              &
     &              NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif

      write (*,'(//a,i8,// )') '### FINAL COLOR NUMBER', NCOLORtot
```

Reordering

NCOLORtot > 1: Multicolor
NCOLORtot = 0: CM
NCOLORtot =-1: RCM
NCOLORtot <-1: CM-RCM

poi_gen (4/9)

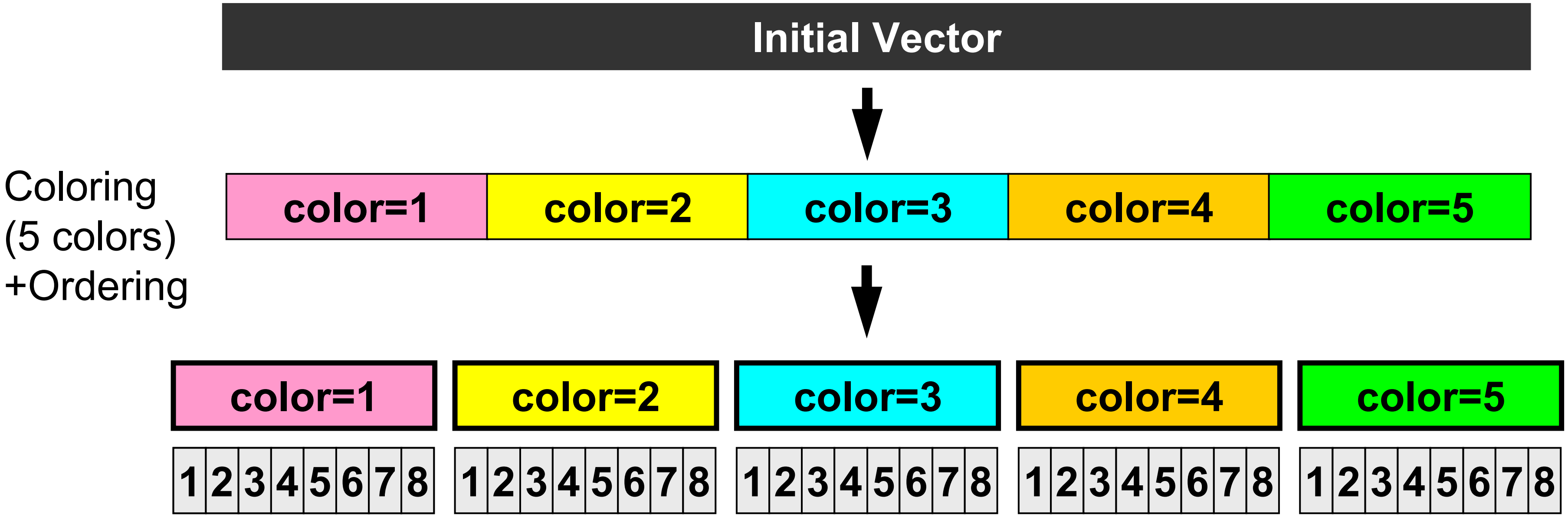
```
      allocate (SMPindex(0:PEsmpTOT*NCOLORtot))
      SMPindex= 0
      do ic= 1, NCOLORtot
        nn1= COLORindex(ic) - COLORindex(ic-1)
        num= nn1 / PEsmpTOT
        nr = nn1 - PEsmpTOT*num
        do ip= 1, PEsmpTOT
          if (ip.le.nr) then
            SMPindex((ic-1)*PEsmpTOT+ip)= num + 1
           else
            SMPindex((ic-1)*PEsmpTOT+ip)= num
          endif
        enddo
      enddo

      do ic= 1, NCOLORtot
        do ip= 1, PEsmpTOT
          j1= (ic-1)*PEsmpTOT + ip
          j0= j1 - 1
          SMPindex(j1)= SMPindex(j0) + SMPindex(j1)
        enddo
      enddo

      allocate (SMPindexG(0:PEsmpTOT))
      SMPindexG= 0
      nn= ICELTOT / PEsmpTOT
      nr= ICELTOT - nn*PEsmpTOT
      do ip= 1, PEsmpTOT
        SMPindexG(ip)= nn
        if (ip.le.nr) SMPindexG(ip)= nn + 1
      enddo

      do ip= 1, PEsmpTOT
        SMPindexG(ip)= SMPindexG(ip-1) + SMPindexG(ip)
      enddo

!C===
```

SMPindex:
for preconditioning

```
  do ic= 1, NCOLORtot
!$omp parallel do …
    do ip= 1, PEsmpTOT
      ip1= (ic-1)*PEsmpTOT+ip
    do i= SMPindex(ip1-1)+1, SMPindex(ip1)
      (…)
    enddo
    enddo
!omp end parallel do
  enddo
```

SMPindex:
for preconditioning

```
  do ic= 1, NCOLORtot
!$omp parallel do …
    do ip= 1, PEsmpTOT
      ip1= (ic-1)*PEsmpTOT+ip
    do i= SMPindex(ip1-1)+1, SMPindex(ip1)
      (…)
    enddo
    enddo
!omp end parallel do
  enddo
```



- 5-colors, 8-threads
- Meshes in same color are independent: parallel processing
- Reordering in ascending order according to color ID

poi_gen (4/9)

```
      allocate (SMPindex(0:PEsmpTOT*NCOLORtot))
      SMPindex= 0
      do ic= 1, NCOLORtot
        nn1= COLORindex(ic) - COLORindex(ic-1)
        num= nn1 / PEsmpTOT
        nr = nn1 - PEsmpTOT*num
        do ip= 1, PEsmpTOT
          if (ip.le.nr) then
            SMPindex((ic-1)*PEsmpTOT+ip)= num + 1
           else
            SMPindex((ic-1)*PEsmpTOT+ip)= num
          endif
        enddo
      enddo

      do ic= 1, NCOLORtot
        do ip= 1, PEsmpTOT
          j1= (ic-1)*PEsmpTOT + ip
          j0= j1 - 1
          SMPindex(j1)= SMPindex(j0) + SMPindex(j1)
        enddo
      enddo

      allocate (SMPindexG(0:PEsmpTOT))
      SMPindexG= 0
      nn= ICELTOT / PEsmpTOT
      nr= ICELTOT - nn*PEsmpTOT
      do ip= 1, PEsmpTOT
        SMPindexG(ip)= nn
        if (ip.le.nr) SMPindexG(ip)= nn + 1
      enddo

      do ip= 1, PEsmpTOT
        SMPindexG(ip)= SMPindexG(ip-1) + SMPindexG(ip)
      enddo

!C===
```

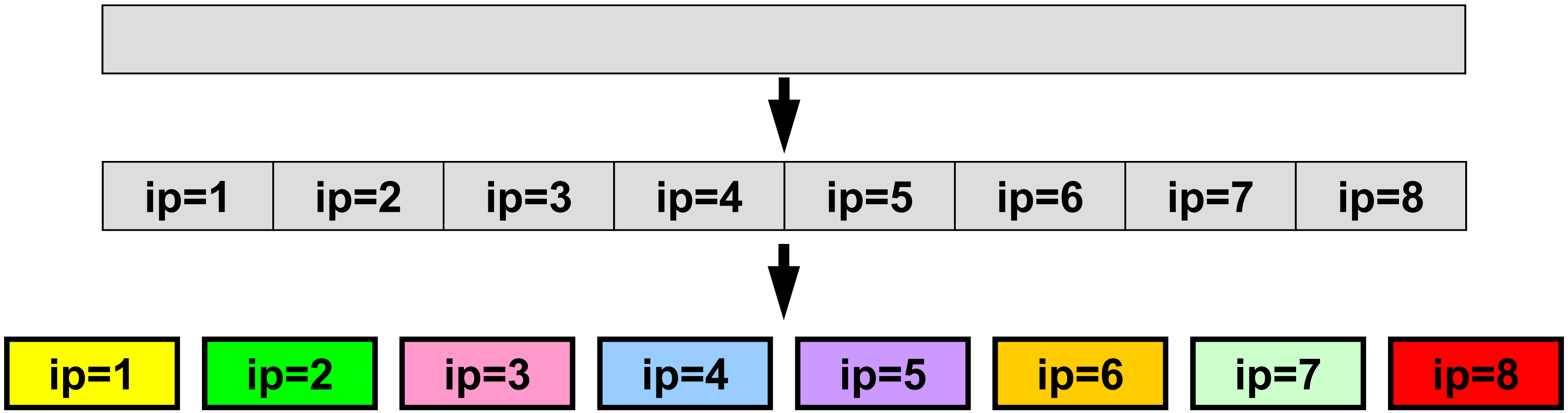
```
!$omp parallel do …
  do ip= 1, PEsmpTOT
  do i= SMPindexG(ip-1)+1, SMPindexG(ip)
     (…)
  enddo
  enddo
!$omp end parallel do
```

SMPindexG:

for Dot-products, DAXPY, Mat-vec, and Poi-gen

SMPindexG

```
!$omp parallel do …
  do ip= 1, PEsmpTOT
  do i= SMPindexG(ip-1)+1, SMPindexG(ip)
    (…)
  enddo
  enddo
!$omp end parallel do
```



for Dot-products, DAXPY, Mat-vec, and Poi-gen

poi_gen (5/9)

New numbering is applied after this point

```
!C
!C-- 1D array
      nn = ICELTOT
      allocate (indexL(0:nn), indexU(0:nn))
      indexL= 0
      indexU= 0

      do icel= 1, ICELTOT
        indexL(icel)= INL(icel)
        indexU(icel)= INU(icel)
      enddo

      do icel= 1, ICELTOT
        indexL(icel)= indexL(icel) + indexL(icel-1)
        indexU(icel)= indexU(icel) + indexU(icel-1)
      enddo

      NPL= indexL(ICELTOT)
      NPU= indexU(ICELTOT)

      allocate (itemL(NPL), AL(NPL))
      allocate (itemU(NPU), AU(NPU))

      itemL= 0
      itemU= 0
        AL= 0.d0
        AU= 0.d0
!C===
```

```
do i= 1, N

  VAL= D(i)*p(i)

  do k= indexL(i-1)+1, indexL(i)
    VAL= VAL + AL(k)*p(itemL(k))
  enddo

  do k= indexU(i-1)+1, indexU(i)
    VAL= VAL + AU(k)*p(itemU(k))
  enddo

  q(i)= VAL

enddo
```

Name	Type	Content
D(N)	**R**	Diagonal components of the matrix (N= ICELTOT)
BFORCE(N)	**R**	RHS vector
PHI(N)	**R**	Unknown vector
indexL(0:N), indexU(0:N)	**I**	# of L/U non-zero off-diag. comp. (CRS)
NPL, NPU	**I**	Total # of L/U non-zero off-diag. comp. (CRS)
itemL(NPL), itemU(NPU)	**I**	Column ID of L/U non-zero off-diag. comp. (CRS)
AL(NPL), AU(NPU)	**R**	L/U non-zero off-diag. comp. (CRS)

poi_gen (6/9)

New numbering is applied

```
!C
!C +-----------------------------------+
!C | INTERIOR & NEUMANN BOUNDARY CELLs |
!C +-----------------------------------+
!C===

!$omp parallel do private
(ip, icel, ic0, icN1, icN2, icN3, icN4, icN5, icN6)   &
!$omp&                 private (VOL0, coef, j, ii, jj, kk)

      do ip   = 1, PEsmpTOT
      do icel= SMPindexG(ip-1)+1, SMPindexG(ip)
        ic0 = NEWtoOLD(icel)

        icN1= NEIBcell(ic0, 1)
        icN2= NEIBcell(ic0, 2)
        icN3= NEIBcell(ic0, 3)
        icN4= NEIBcell(ic0, 4)
        icN5= NEIBcell(ic0, 5)
        icN6= NEIBcell(ic0, 6)

        VOL0= VOLCEL  (ic0)

        if (icN5.ne.0) then
          icN5= OLDtoNEW(icN5)
          coef= RDZ * ZAREA
          D(icel)= D(icel) - coef

          if (icN5.lt.icel) then
            do j= 1, INL(icel)
              if (IAL(j,icel).eq.icN5) then
                itemL(j+indexL(icel-1))= icN5
                   AL(j+indexL(icel-1))= coef
                exit
              endif
            enddo
           else
            do j= 1, INU(icel)
              if (IAU(j,icel).eq.icN5) then
                itemU(j+indexU(icel-1))= icN5
                   AU(j+indexU(icel-1))= coef
                exit
              endif
            enddo
          endif
        endif
```

icel: New ID
ic0: Old ID

$$\frac{\phi_{neib(icel,1)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+$$

$$\frac{\phi_{neib(icel,2)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+$$

$$\frac{\phi_{neib(icel,3)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+$$

$$\frac{\phi_{neib(icel,4)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+$$

$$\frac{\phi_{neib(icel,5)}-\phi_{icel}}{\Delta z}\Delta x\Delta y+$$

$$\frac{\phi_{neib(icel,6)}-\phi_{icel}}{\Delta z}\Delta x\Delta y=f_{icel}\Delta x\Delta y\Delta z$$

Coef. Matrix: Parallel, “SMPindexG”

“private”

```
!C
!C +-----------------------------------+
!C | INTERIOR & NEUMANN BOUNDARY CELLs |
!C +-----------------------------------+
!C===

!$omp parallel do private (ip,icel,ic0,icN1,icN2,icN3,icN4,icN5,icN6)  &
!$omp&            private (VOL0,coef,j,ii,jj,kk)

      do ip   = 1, PEsmpTOT
      do icel= SMPindexG(ip-1)+1, SMPindexG(ip)
        ic0 = NEWtoOLD(icel)

        icN1= NEIBcell(ic0,1)
        icN2= NEIBcell(ic0,2)
        icN3= NEIBcell(ic0,3)
        icN4= NEIBcell(ic0,4)
        icN5= NEIBcell(ic0,5)
        icN6= NEIBcell(ic0,6)

        VOL0= VOLCEL  (ic0)
…
```

poi_gen (6/9)

New numbering is applied

```
!C
!C +-----------------------------------+
!C | INTERIOR & NEUMANN BOUNDARY CELLs |
!C +-----------------------------------+
!C===

!$omp parallel do private
(ip, icel, ic0, icN1, icN2, icN3, icN4, icN5, icN6)   &
!$omp&                 private (VOL0, coef, j, ii, jj, kk)

      do ip  = 1, PEsmpTOT
      do icel= SMPindexG(ip-1)+1, SMPindexG(ip)
        ic0 = NEWtoOLD(icel)

        icN1= NEIBcell(ic0, 1)
        icN2= NEIBcell(ic0, 2)
        icN3= NEIBcell(ic0, 3)
        icN4= NEIBcell(ic0, 4)
        icN5= NEIBcell(ic0, 5)
        icN6= NEIBcell(ic0, 6)

        VOL0= VOLCEL  (ic0)

        if (icN5.ne.0) then
          icN5= OLDtoNEW(icN5)
          coef= RDZ * ZAREA
          D(icel)= D(icel) - coef

          if (icN5.lt.icel) then
            do j= 1, INL(icel)
              if (IAL(j,icel).eq.icN5) then
                itemL(j+indexL(icel-1))= icN5
                   AL(j+indexL(icel-1))= coef
                exit
              endif
            enddo
           else
            do j= 1, INU(icel)
              if (IAU(j,icel).eq.icN5) then
                itemU(j+indexU(icel-1))= icN5
                   AU(j+indexU(icel-1))= coef
                exit
              endif
            enddo
          endif
        endif
```

icel: New ID
ic0: Old ID

$$\frac{\phi_{neib(icel,1)} - \phi_{icel}}{\Delta x}\Delta y \Delta z + \frac{\phi_{neib(icel,2)} - \phi_{icel}}{\Delta x}\Delta y \Delta z + \frac{\phi_{neib(icel,3)} - \phi_{icel}}{\Delta y}\Delta z \Delta x + \frac{\phi_{neib(icel,4)} - \phi_{icel}}{\Delta y}\Delta z \Delta x + \frac{\phi_{neib(icel,5)} - \phi_{icel}}{\Delta z}\Delta x \Delta y + \frac{\phi_{neib(icel,6)} - \phi_{icel}}{\Delta z}\Delta x \Delta y = f_{icel}\Delta x \Delta y \Delta z$$

poi_gen (6/9)

New numbering is applied

```
!C
!C +-------------------------------------+
!C | INTERIOR & NEUMANN BOUNDARY CELLs |
!C +-------------------------------------+
!C===

!$omp parallel do private
(ip,icel,ic0,icN1,icN2,icN3,icN4,icN5,icN6)   &
!$omp&             private (VOL0,coef,j,ii,jj,kk)

      do ip   = 1, PEsmpTOT
      do icel= SMPindexG(ip-1)+1, SMPindexG(ip)
        ic0 = NEWtoOLD(icel)

        icN1= NEIBcell(ic0,1)
        icN2= NEIBcell(ic0,2)
        icN3= NEIBcell(ic0,3)
        icN4= NEIBcell(ic0,4)
        icN5= NEIBcell(ic0,5)
        icN6= NEIBcell(ic0,6)

        VOL0= VOLCEL  (ic0)

        if (icN5.ne.0) then
          icN5= OLDtoNEW(icN5)
          coef= RDZ * ZAREA
          D(icel)= D(icel) - coef

          if (icN5.lt.icel) then
            do j= 1, INL(icel)
              if (IAL(j,icel).eq.icN5) then
                itemL(j+indexL(icel-1))= icN5
                   AL(j+indexL(icel-1))= coef
                exit
              endif
            enddo
           else
            do j= 1, INU(icel)
              if (IAU(j,icel).eq.icN5) then
                itemU(j+indexU(icel-1))= icN5
                   AU(j+indexU(icel-1))= coef
                exit
              endif
            enddo
          endif
        endif
```

$$\frac{\phi_{neib(icel,1)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+$$

$$\frac{\phi_{neib(icel,2)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+$$

$$\frac{\phi_{neib(icel,3)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+$$

$$\frac{\phi_{neib(icel,4)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+$$

$$\frac{\phi_{neib(icel,5)}-\phi_{icel}}{\Delta z}\Delta x\Delta y+$$

$$\frac{\phi_{neib(icel,6)}-\phi_{icel}}{\Delta z}\Delta x\Delta y=f_{icel}\Delta x\Delta y\Delta z$$

poi_gen (6/9)

New numbering is applied

```
!C
!C +-------------------------------------+
!C | INTERIOR & NEUMANN BOUNDARY CELLs |
!C +-------------------------------------+
!C===

!$omp parallel do private
(ip, icel, ic0, icN1, icN2, icN3, icN4, icN5, icN6)   &
!$omp&            private (VOL0, coef, j, ii, jj, kk)

      do ip   = 1, PEsmpTOT
      do icel= SMPindexG(ip-1)+1, SMPindexG(ip)
        ic0 = NEWtoOLD(icel)

        icN1= NEIBcell(ic0, 1)
        icN2= NEIBcell(ic0, 2)
        icN3= NEIBcell(ic0, 3)
        icN4= NEIBcell(ic0, 4)
        icN5= NEIBcell(ic0, 5)
        icN6= NEIBcell(ic0, 6)

        VOL0= VOLCEL  (ic0)

        if (icN5.ne.0) then
          icN5= OLDtoNEW(icN5)
          coef= RDZ * ZAREA
          D(icel)= D(icel) - coef

          if (icN5.lt.icel) then
            do j= 1, INL(icel)
              if (IAL(j,icel).eq.icN5) then
                itemL(j+indexL(icel-1))= icN5
                   AL(j+indexL(icel-1))= coef
                exit
              endif
            enddo
           else
            do j= 1, INU(icel)
              if (IAU(j,icel).eq.icN5) then
                itemU(j+indexU(icel-1))= icN5
                   AU(j+indexU(icel-1))= coef
                exit
              endif
            enddo
          endif
        endif
```

$$RDZ = \frac{1}{\Delta z}$$

$$ZAREA = \Delta x \Delta y$$

icN5 < icel
Lower Part

$$\frac{\phi_{neib(icel,1)} - \phi_{icel}}{\Delta x} \Delta y \Delta z + \frac{\phi_{neib(icel,2)} - \phi_{icel}}{\Delta x} \Delta y \Delta z + \frac{\phi_{neib(icel,3)} - \phi_{icel}}{\Delta y} \Delta z \Delta x + \frac{\phi_{neib(icel,4)} - \phi_{icel}}{\Delta y} \Delta z \Delta x + \boxed{\frac{\phi_{neib(icel,5)} - \phi_{icel}}{\Delta z} \Delta x \Delta y} + \frac{\phi_{neib(icel,6)} - \phi_{icel}}{\Delta z} \Delta x \Delta y = f_{icel} \Delta x \Delta y \Delta z$$

poi_gen (6/9)

New numbering is applied

```
!C
!C +-----------------------------------+
!C | INTERIOR & NEUMANN BOUNDARY CELLs |
!C +-----------------------------------+
!C===

!$omp parallel do private
(ip,icel,ic0,icN1,icN2,icN3,icN4,icN5,icN6)   &
!$omp&                private (VOL0,coef,j,ii,jj,kk)

      do ip   = 1, PEsmpTOT
      do icel= SMPindexG(ip-1)+1, SMPindexG(ip)
        ic0 = NEWtoOLD(icel)

        icN1= NEIBcell(ic0,1)
        icN2= NEIBcell(ic0,2)
        icN3= NEIBcell(ic0,3)
        icN4= NEIBcell(ic0,4)
        icN5= NEIBcell(ic0,5)
        icN6= NEIBcell(ic0,6)

        VOL0= VOLCEL  (ic0)

        if (icN5.ne.0) then
          icN5= OLDtoNEW(icN5)
          coef= RDZ * ZAREA
          D(icel)= D(icel) - coef

          if (icN5.lt.icel) then
            do j= 1, INL(icel)
              if (IAL(j,icel).eq.icN5) then
                itemL(j+indexL(icel-1))= icN5
                   AL(j+indexL(icel-1))= coef
                exit
              endif
            enddo
           else
            do j= 1, INU(icel)
              if (IAU(j,icel).eq.icN5) then
                itemU(j+indexU(icel-1))= icN5
                   AU(j+indexU(icel-1))= coef
                exit
              endif
            enddo
          endif
        endif
```

$$RDZ = \frac{1}{\Delta z}$$

$$ZAREA = \Delta x \Delta y$$

icN5 > icel
Upper Part

$$\frac{\phi_{neib(icel,1)} - \phi_{icel}}{\Delta x}\Delta y \Delta z + \frac{\phi_{neib(icel,2)} - \phi_{icel}}{\Delta x}\Delta y \Delta z + \frac{\phi_{neib(icel,3)} - \phi_{icel}}{\Delta y}\Delta z \Delta x + \frac{\phi_{neib(icel,4)} - \phi_{icel}}{\Delta y}\Delta z \Delta x + \frac{\phi_{neib(icel,5)} - \phi_{icel}}{\Delta z}\Delta x \Delta y + \boxed{\frac{\phi_{neib(icel,6)} - \phi_{icel}}{\Delta z}\Delta x \Delta y} = f_{icel}\Delta x \Delta y \Delta z$$

poi_gen (7/9)

```
      if (icN3.ne.0) then
        icN3= OLDtoNEW(icN3)
        coef= RDY * YAREA
        D(icel)= D(icel) - coef

        if (icN3.lt.icel) then
          do j= 1, INL(icel)
            if (IAL(j,icel).eq.icN3) then
              itemL(j+indexL(icel-1))= icN3
                 AL(j+indexL(icel-1))= coef
              exit
            endif
          enddo
         else
          do j= 1, INU(icel)
            if (IAU(j,icel).eq.icN3) then
              itemU(j+indexU(icel-1))= icN3
                 AU(j+indexU(icel-1))= coef
              exit
            endif
          enddo
        endif
      endif

      if (icN1.ne.0) then
        icN1= OLDtoNEW(icN1)
        coef= RDX * XAREA
        D(icel)= D(icel) - coef

        if (icN1.lt.icel) then
          do j= 1, INL(icel)
            if (IAL(j,icel).eq.icN1) then
              itemL(j+indexL(icel-1))= icN1
                 AL(j+indexL(icel-1))= coef
              exit
            endif
          enddo
         else
          do j= 1, INU(icel)
            if (IAU(j,icel).eq.icN1) then
              itemU(j+indexU(icel-1))= icN1
                 AU(j+indexU(icel-1))= coef
              exit
            endif
          enddo
        endif
      endif
```

$$
\frac{\phi_{neib(icel,1)} - \phi_{icel}}{\Delta x} \Delta y \Delta z +
$$

$$
\frac{\phi_{neib(icel,2)} - \phi_{icel}}{\Delta x} \Delta y \Delta z +
$$

$$
\frac{\phi_{neib(icel,3)} - \phi_{icel}}{\Delta y} \Delta z \Delta x +
$$

$$
\frac{\phi_{neib(icel,4)} - \phi_{icel}}{\Delta y} \Delta z \Delta x +
$$

$$
\frac{\phi_{neib(icel,5)} - \phi_{icel}}{\Delta z} \Delta x \Delta y +
$$

$$
\frac{\phi_{neib(icel,6)} - \phi_{icel}}{\Delta z} \Delta x \Delta y = f_{icel} \Delta x \Delta y \Delta z
$$

poi_gen (8/9)

```
      if (icN2.ne.0) then
        icN2= OLDtoNEW(icN2)
        coef= RDX * XAREA
        D(icel)= D(icel) - coef

        if (icN2.lt.icel) then
          do j= 1, INL(icel)
            if (IAL(j,icel).eq.icN2) then
              itemL(j+indexL(icel-1))= icN2
                 AL(j+indexL(icel-1))= coef
              exit
            endif
          enddo
         else
          do j= 1, INU(icel)
            if (IAU(j,icel).eq.icN2) then
              itemU(j+indexU(icel-1))= icN2
                 AU(j+indexU(icel-1))= coef
              exit
            endif
          enddo
        endif
      endif

      if (icN4.ne.0) then
        icN4= OLDtoNEW(icN4)
        coef= RDY * YAREA
        D(icel)= D(icel) - coef

        if (icN4.lt.icel) then
          do j= 1, INL(icel)
            if (IAL(j,icel).eq.icN4) then
              itemL(j+indexL(icel-1))= icN4
                 AL(j+indexL(icel-1))= coef
              exit
            endif
          enddo
         else
          do j= 1, INU(icel)
            if (IAU(j,icel).eq.icN4) then
              itemU(j+indexU(icel-1))= icN4
                 AU(j+indexU(icel-1))= coef
              exit
            endif
          enddo
        endif
      endif
```

$$\frac{\phi_{neib(icel,1)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+$$
$$\frac{\phi_{neib(icel,2)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+$$
$$\frac{\phi_{neib(icel,3)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+$$
$$\frac{\phi_{neib(icel,4)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+$$
$$\frac{\phi_{neib(icel,5)}-\phi_{icel}}{\Delta z}\Delta x\Delta y+$$
$$\frac{\phi_{neib(icel,6)}-\phi_{icel}}{\Delta z}\Delta x\Delta y= f_{icel}\Delta x\Delta y\Delta z$$

poi_gen (9/9)

```fortran
!$omp parallel do private
(ip,icel,ic0,icN1,icN2,icN3,icN4,icN5,icN6)   &
!$omp&              private (VOL0,coef,j,ii,jj,kk)

…

        if (icN6.ne.0) then
          icN6= OLDtoNEW(icN6)
          coef= RDZ * ZAREA
          D(icel)= D(icel) - coef

          if (icN6.lt.icel) then
            do j= 1, INL(icel)
              if (IAL(j,icel).eq.icN6) then
                itemL(j+indexL(icel-1))= icN6
                   AL(j+indexL(icel-1))= coef
                exit
              endif
            enddo
           else
            do j= 1, INU(icel)
              if (IAU(j,icel).eq.icN6) then
                itemU(j+indexU(icel-1))= icN6
                   AU(j+indexU(icel-1))= coef
                exit
              endif
            enddo
          endif
        endif
        ii= XYZ(ic0,1)
        jj= XYZ(ic0,2)
        kk= XYZ(ic0,3)

        BFORCE(icel)= -dfloat(ii+jj+kk) * VOL0

      enddo
      enddo
!$omp end parallel do
!C===
```

BFORCE using original mesh ID

ii,jj,kk,VOL0: private

$$
\frac{\phi_{neib(icel,1)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+
$$
$$
\frac{\phi_{neib(icel,2)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+
$$
$$
\frac{\phi_{neib(icel,3)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+
$$
$$
\frac{\phi_{neib(icel,4)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+
$$
$$
\frac{\phi_{neib(icel,5)}-\phi_{icel}}{\Delta z}\Delta x\Delta y+
$$
$$
\frac{\phi_{neib(icel,6)}-\phi_{icel}}{\Delta z}\Delta x\Delta y = f_{icel}\Delta x\Delta y\Delta z
$$

Main Program

```
      program MAIN

      use STRUCT
      use PCG
      use solver_ICCG_mc

      implicit REAL*8 (A-H,O-Z)
      real(kind=8), dimension(:), allocatable :: WK

      call INPUT
      call POINTER_INIT
      call BOUNDARY_CELL
      call CELL_METRICS
      call POI_GEN

      PHI=  0.d0

        call solve_ICCG_mc                                              &
     &       ( ICELTOT, NPL, NPU, indexL, itemL, indexU, itemU, D,       &
     &         BFORCE,  PHI, AL, AU, NCOLORtot, PEsmpTOT,                &
     &         SMPindex, SMPindexG, EPSICCG, ITR, IER)
```

solve_ICCG_mc (1/6)

```
!C***
!C*** module solver_ICCG_mc
!C***
!
      module solver_ICCG_mc
      contains
!C
!C*** solve_ICCG
!C
subroutine solve_ICCG_mc                                              &
     &           ( N, NPL, NPU, indexL, itemL, indexU, itemU, D, B, X,    &
     &             AL, AU, NCOLORtot, PEsmpTOT, SMPindex, SMPindexG,      &
     &             EPS, ITR, IER)

      implicit REAL*8 (A-H,O-Z)

      integer :: N, NL, NU, NCOLORtot, PEsmpTOT

      real(kind=8), dimension(N)   :: D
      real(kind=8), dimension(N)   :: B
      real(kind=8), dimension(N)   :: X

      real(kind=8), dimension(NPL) :: AL
      real(kind=8), dimension(NPU) :: AU

      integer, dimension(0:N)   :: indexL, indexU
      integer, dimension(NPL):: itemL
      integer, dimension(NPU):: itemU

      integer, dimension(0:NCOLORtot*PEsmpTOT):: SMPindex
      integer, dimension(0:PEsmpTOT)          :: SMPindexG

      real(kind=8), dimension(:,:), allocatable :: W

      integer, parameter ::  R= 1
      integer, parameter ::  Z= 2
      integer, parameter ::  Q= 2
      integer, parameter ::  P= 3
      integer, parameter :: DD= 4
```

solve_ICCG_mc (2/6)

```
!C
!C +-------+
!C | INIT |
!C +-------+
!C===
      allocate (W(N+128,4))

!$omp parallel do private(ip,i)
      do ip= 1, PEsmpTOT
      do i = SMPindexG(ip-1)+1, SMPindexG(ip)
        X(i)   = 0.d0
        W(i,2)= 0.0D0
        W(i,3)= 0.0D0
        W(i,4)= 0.0D0
      enddo
      enddo
!$omp end parallel do


      do ic= 1, NCOLORtot
!$omp parallel do private(ip,ip1,i,VAL,k)
        do ip= 1, PEsmpTOT
          ip1= (ic-1)*PEsmpTOT + ip
        do i= SMPindex(ip1-1)+1, SMPindex(ip1)
          VAL= D(i)
          do k= indexL(i-1)+1, indexL(i)
            VAL= VAL - (AL(k)**2) *
W(itemL(k),DD)
          enddo
          W(i,DD)= 1.d0/VAL
        enddo
        enddo
!$omp end parallel do
      enddo
```

Incomplete “Modified” Cholesky Factorization

Incomplete "Modified" Cholesky Factorization

$$d_i = \left(a_{ii} - \sum_{k=1}^{i-1} {a_{ik}}^2 \cdot d_k \right)^{-1} = {l_{ii}}^{-1}$$

W[DD][i]:	d_i
D[i]:	a_{ii}
itemL[j]:	k
AL[j]:	a_{ik}

```
do i= 1, N
  VAL= D(i)
  do k= indexL(i-1)+1, indexL(i)
    VAL= VAL - (AL(k)**2) * W(itemL(k),DD)
  enddo
  W(i,DD)= 1.d0/VAL
enddo
```

Incomplete "Modified" Cholesky Factorization: Parallel Version

$$d_i = \left(a_{ii} - \sum_{k=1}^{i-1} a_{ik}{}^2 \cdot d_k \right)^{-1} = l_{ii}{}^{-1}$$

W[DD][i]:	d_i
D[i]:	a_{ii}
itemL[j]:	k
AL[j]:	a_{ik}

```
      do ic= 1, NCOLORtot
!$omp parallel do private(ip,ip1,i,VAL,k)
        do ip= 1, PEsmpTOT
          ip1= (ic-1)*PEsmpTOT + ip
        do i= SMPindex(ip1-1)+1, SMPindex(ip1)
          VAL= D(i)
          do k= indexL(i-1)+1, indexL(i)
            VAL= VAL - (AL(k)**2) * W(itemL(k),DD)
          enddo
          W(i,DD)= 1.d0/VAL
        enddo
        enddo
!$omp end parallel do
      enddo
```

solve_ICCG_mc (3/6)

```
!C +-----------------------+
!C | {r0}= {b} - [A]{xini} |
!C +-----------------------+
!C===
!$omp parallel do private(ip,i,VAL,k)
        do ip= 1, PEsmpTOT
        do i = SMPindexG(ip-1)+1,
     &         SMPindexG(ip)
          VAL= D(i)*X(i)
          do k= indexL(i-1)+1, indexL(i)
            VAL= VAL + AL(k)*X(itemL(k))
          enddo
          do k= indexU(i-1)+1, indexU(i)
            VAL= VAL + AU(k)*X(itemU(k))
          enddo
          W(i,R)= B(i) - VAL
        enddo
        enddo
!$omp end parallel do

      BNRM2= 0.0D0
!$omp parallel do private(ip,i)
reduction(+:BNRM2)
      do ip= 1, PEsmpTOT
      do i = SMPindexG(ip-1)+1, SMPindexG(ip)
        BNRM2 = BNRM2 + B(i)  **2
      enddo
      enddo
!$omp end parallel do
!C===
```

Compute $r^{(0)}= b-[A]x^{(0)}$

for i= 1, 2, …
- solve $[M]z^{(i-1)}= r^{(i-1)}$
- $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
- **if** i=1
  - $p^{(1)}= z^{(0)}$
- **else**
  - $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
  - $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
- **endif**
- $q^{(i)}= [A]p^{(i)}$
- $\alpha_i\ = \rho_{i-1}/p^{(i)}q^{(i)}$
- $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
- $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
- check convergence |r|

end

Mat-Vec

NO Data Dependency: SMPindexG

```
!$omp parallel do private(ip,i,VAL,k)
      do ip= 1, PEsmpTOT
        do i = SMPindexG(ip-1)+1, SMPindexG(ip)
          VAL= D(i)*X(i)
          do k= indexL(i-1)+1, indexL(i)
            VAL= VAL + AL(k)*X(itemL(k))
          enddo
          do k= indexU(i-1)+1, indexU(i)
            VAL= VAL + AU(k)*X(itemU(k))
          enddo
          W(i,R)= B(i) - VAL
        enddo
      enddo
!$omp end parallel do
```

solve_ICCG_mc (3/6)

```
!C +-----------------------+
!C | {r0}= {b} - [A]{xini} |
!C +-----------------------+
!C===
!$omp parallel do private(ip,i,VAL,k)
        do ip= 1, PEsmpTOT
        do i = SMPindexG(ip-1)+1, SMPindexG(ip)
          VAL= D(i)*X(i)
          do k= indexL(i-1)+1, indexL(i)
            VAL= VAL + AL(k)*X(itemL(k))
          enddo
          do k= indexU(i-1)+1, indexU(i)
            VAL= VAL + AU(k)*X(itemU(k))
          enddo
          W(i,R)= B(i) - VAL
        enddo
        enddo
!$omp end parallel do

      BNRM2= 0.0D0
!$omp parallel do private(ip,i) reduction(+:BNRM2)
      do ip= 1, PEsmpTOT
      do i = SMPindexG(ip-1)+1, SMPindexG(ip)
        BNRM2 = BNRM2 + B(i)  **2
      enddo
      enddo
!$omp end parallel do
!C===
```

Compute $r^{(0)}= b-[A]x^{(0)}$

for i= 1, 2, …
- solve $[M]z^{(i-1)}= r^{(i-1)}$
- $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
- **if** i=1
  - $p^{(1)}= z^{(0)}$
- **else**
  - $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
  - $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
- **endif**
- $q^{(i)}= [A]p^{(i)}$
- $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
- $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
- $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
- check convergence |r|

end

Dot Products: SMPindexG, reduction

```
      BNRM2= 0.0D0
!$omp parallel do private(ip,i) reduction(+:BNRM2)
      do ip= 1, PEsmpTOT
      do i = SMPindexG(ip-1)+1, SMPindexG(ip)
        BNRM2 = BNRM2 + B(i)  **2
      enddo
      enddo
!$omp end parallel do
```

solve_ICCG_mc (4/6)

```
      ITR= N
      do L= 1, ITR
!C
!C +----------------+
!C | {z}= [Minv]{r} |
!C +----------------+
!C===
!$omp parallel do private(ip,i)
      do ip= 1, PEsmpTOT
      do i = SMPindexG(ip-1)+1, SMPindexG(ip)
        W(i,Z)= W(i,R)
      enddo
      enddo
!$omp end parallel do

      do ic= 1, NCOLORtot
!$omp parallel do private(ip,ip1,i,WVAL,j)
        do ip= 1, PEsmpTOT
          ip1= (ic-1)*PEsmpTOT + ip
        do i= SMPindex(ip1-1)+1, SMPindex(ip1)
          WVAL= W(i,Z)
          do j= 1, INL(i)
            WVAL= WVAL -  AL(j,i) * W(IAL(j,i),Z)
          enddo
          W(i,Z)= WVAL * W(i,DD)
        enddo
        enddo
!$omp end parallel do
      enddo

      do ic= NCOLORtot, 1, -1
!$omp parallel do private(ip,ip1,i,SW,j)
        do ip= 1, PEsmpTOT
          ip1= (ic-1)*PEsmpTOT + ip
        do i= SMPindex(ip1-1)+1, SMPindex(ip1)
          SW  = 0.0d0
          do j= 1, INU(i)
            SW= SW + AU(j,i) * W(IAU(j,i),Z)
          enddo
          W(i,Z)= W(i,Z) - W(i,DD) * SW
        enddo
        enddo
!$omp end parallel do
      enddo
!C===
```

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    solve [M]z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

solve_ICCG_mc (4/6)

```
      ITR= N
      do L= 1, ITR
!C
!C +----------------+
!C | {z}= [Minv]{r} |
!C +----------------+
!C===
!$omp parallel do private(ip,i)
      do ip= 1, PEsmpTOT
      do i = SMPindexG(ip-1)+1, SMPindexG(ip)
        W(i,Z)= W(i,R)
      enddo
      enddo
!$omp end parallel do

      do ic= 1, NCOLORtot
!$omp parallel do private(ip,ip1,i,WVAL,k)
        do ip= 1, PEsmpTOT
          ip1= (ic-1)*PEsmpTOT + ip
        do i= SMPindex(ip1-1)+1, SMPindex(ip1)
          WVAL= W(i,Z)
          do k= indexL(i-1)+1, indexL(i)
            WVAL= WVAL -  AL(k) * W(itemL(k),Z)
          enddo
          W(i,Z)= WVAL * W(i,DD)
        enddo
        enddo
!$omp end parallel do
      enddo

      do ic= NCOLORtot, 1, -1
!$omp parallel do private(ip,ip1,i,SW,k)
        do ip= 1, PEsmpTOT
          ip1= (ic-1)*PEsmpTOT + ip
        do i= SMPindex(ip1-1)+1, SMPindex(ip1)
          SW  = 0.0d0
          do k= indexU(i-1)+1, indexU(i)
            SW= SW + AU(k) * W(itemU(k),Z)
          enddo
          W(i,Z)= W(i,Z) - W(i,DD) * SW
        enddo
        enddo
!$omp end parallel do
      enddo
!C===
```

SMPindex

Compute $r^{(0)}= b-[A]x^{(0)}$
<u>for</u> i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  <u>if</u> i=1
    $p^{(1)}= z^{(0)}$
   <u>else</u>
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
  <u>endif</u>
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i\ = \rho_{i-1}/p^{(i)}q^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence |r|
<u>end</u>

solve_ICCG_mc (4/6)

SMPindex

```
      ITR= N
      do L= 1, ITR
!C
!C +----------------+
!C | {z}= [Minv]{r} |
!C +----------------+
!C===
!$omp parallel do private(ip,i)
      do ip= 1, PEsmpTOT
      do i = SMPindexG(ip-1)+1, SMPindexG(ip)
        W(i,Z)= W(i,R)
      enddo
      enddo
!$omp end parallel do
 do ic= 1, NCOLORtot
!$omp parallel do private(ip,ip1,i,WVAL,k)
        do ip= 1, PEsmpTOT
          ip1= (ic-1)*PEsmpTOT + ip
        do i= SMPindex(ip1-1)+1, SMPindex(ip1)
          WVAL= W(i,Z)
          do k= indexL(i-1)+1, indexL(i)
            WVAL= WVAL -  AL(k) * W(itemL(k),Z)
          enddo
          W(i,Z)= WVAL * W(i,DD)
        enddo
        enddo
!$omp end parallel do
      enddo

      do ic= NCOLORtot, 1, -1
!$omp parallel do private(ip,ip1,i,SW,k)
        do ip= 1, PEsmpTOT
          ip1= (ic-1)*PEsmpTOT + ip
        do i= SMPindex(ip1-1)+1, SMPindex(ip1)
          SW  = 0.0d0
          do k= indexU(i-1)+1, indexU(i)
            SW= SW + AU(k) * W(itemU(k),Z)
          enddo
          W(i,Z)= W(i,Z) - W(i,DD) * SW
        enddo
        enddo
!$omp end parallel do
      enddo
!C===
```

$$(M)\{z\} = (LDL^T)\{z\} = \{r\}$$

$$(L)\{z\} = \{r\}$$

Forward Substitution

$$(DL^T)\{z\} = \{z\}$$

Backward Substitution

Forward Substitution: SMPindex

```
      do ic= 1, NCOLORtot
!$omp parallel do private(ip,ip1,i,WVAL,k)
        do ip= 1, PEsmpTOT
          ip1= (ic-1)*PEsmpTOT + ip
        do i= SMPindex(ip1-1)+1, SMPindex(ip1)
          WVAL= W(i,Z)
          do k= indexL(i-1)+1, indexL(i)
            WVAL= WVAL -  AL(k) * W(indexL(k),Z)
          enddo
          W(i,Z)= WVAL * W(i,DD)
        enddo
        enddo
!$omp end parallel do
      enddo
```

solve_ICCG_mc (5/6)

```
!C +-----------------------------+
!C | {p} = {z} if      ITER=1    |
!C | BETA= RHO / RHO1  otherwise |
!C +-----------------------------+
!C===
      if ( L.eq.1 ) then
!$omp parallel do private(ip,i)
        do ip= 1, PEsmpTOT
        do i = SMPindexG(ip-1)+1, SMPindexG(ip)
          W(i,P)= W(i,Z)
        enddo
        enddo
!$omp end parallel do
       else
        BETA= RHO / RHO1
!$omp parallel do private(ip,i)
        do ip= 1, PEsmpTOT
        do i = SMPindexG(ip-1)+1, SMPindexG(ip)
          W(i,P)= W(i,Z) + BETA*W(i,P)
        enddo
        enddo
!$omp end parallel do
      endif
!C===

!C +-------------+
!C | {q}= [A]{p} |
!C +-------------+
!C===
!$omp parallel do private(ip,i,VAL,k)
      do ip= 1, PEsmpTOT
        do i = SMPindexG(ip-1)+1, SMPindexG(ip)
          VAL= D(i)*W(i,P)
          do k= indexL(i-1)+1, indexL(i)
            VAL= VAL + AL(k)*W(itemL(k),P)
          enddo
          do k= indexU(i-1)+1, indexU(i)
            VAL= VAL + AU(k)*W(itemU(k),P)
          enddo
          W(i,Q)= VAL
        enddo
      enddo
!$omp end parallel do
!C===
```

Compute $r^{(0)}= b-[A]x^{(0)}$

for i= 1, 2, …

  solve $[M]z^{(i-1)}= r^{(i-1)}$

  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$

  if i=1

   $p^{(1)}= z^{(0)}$

  else

   $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$

   $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$

  endif

  $q^{(i)}= [A]p^{(i)}$

  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$

  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$

  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$

  check convergence |r|

end

solve_ICCG_mc (5/6)

```
!C +-----------------------------+
!C | {p} = {z} if      ITER=1    |
!C | BETA= RHO / RHO1  otherwise |
!C +-----------------------------+
!C===
      if ( L.eq.1 ) then
!$omp parallel do private(ip,i)
        do ip= 1, PEsmpTOT
        do i = SMPindexG(ip-1)+1, SMPindexG(ip)
          W(i,P)= W(i,Z)
        enddo
        enddo
!$omp end parallel do
       else
        BETA= RHO / RHO1
!$omp parallel do private(ip,i)
        do ip= 1, PEsmpTOT
        do i = SMPindexG(ip-1)+1, SMPindexG(ip)
          W(i,P)= W(i,Z) + BETA*W(i,P)
        enddo
        enddo
!$omp end parallel do
      endif
!C===

!C +-------------+
!C | {q}= [A]{p} |
!C +-------------+
!C===
!$omp parallel do private(ip,i,VAL,k)
      do ip= 1, PEsmpTOT
        do i = SMPindexG(ip-1)+1, SMPindexG(ip)
          VAL= D(i)*W(i,P)
          do k= indexL(i-1)+1, indexL(i)
            VAL= VAL + AL(k)*W(itemL(k),P)
          enddo
          do k= indexU(i-1)+1, indexU(i)
            VAL= VAL + AU(k)*W(itemU(k),P)
          enddo
          W(i,Q)= VAL
        enddo
      enddo
!$omp end parallel do
!C===
```

Compute $r^{(0)}= b-[A]x^{(0)}$

<u>for</u> i= 1, 2, …

    solve $[M]z^{(i-1)}= r^{(i-1)}$

    $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$

    <u>if</u> i=1

      $p^{(1)}= z^{(0)}$

     <u>else</u>

      $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$

      $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$

    <u>endif</u>

    $q^{(i)}= [A]p^{(i)}$

    $\alpha_i\ = \rho_{i-1}/p^{(i)}q^{(i)}$

    $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$

    $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$

    check convergence |r|

<u>end</u>

solve_ICCG_mc (6/6)

```
!C
!C +----------------------+
!C | ALPHA= RHO / {p}{q}  |
!C +----------------------+
!C===
      C1= 0.d0
!$omp parallel do private(ip,i) reduction(+:C1)
      do ip= 1, PEsmpTOT
      do i = SMPindexG(ip-1)+1, SMPindexG(ip)
        C1= C1 + W(i,P)*W(i,Q)
      enddo
      enddo
!$omp end parallel do

      ALPHA= RHO / C1
!C===


!C
!C +----------------------+
!C | {x}= {x} + ALPHA*{p} |
!C | {r}= {r} - ALPHA*{q} |
!C +----------------------+
!C===
!$omp parallel do private(ip,i)
      do ip= 1, PEsmpTOT
      do i = SMPindexG(ip-1)+1, SMPindexG(ip)
        X(i)   = X(i)    + ALPHA * W(i,P)
        W(i,R)= W(i,R) - ALPHA * W(i,Q)
      enddo
      enddo
!$omp end parallel do

      DNRM2= 0.d0
!$omp parallel do private(ip,i) reduction(+:DNRM2)
      do ip= 1, PEsmpTOT
      do i = SMPindexG(ip-1)+1, SMPindexG(ip)
        DNRM2= DNRM2 + W(i,R)**2
      enddo
      enddo
!$omp end parallel do
!C===
```

Compute $r^{(0)}= b-[A]x^{(0)}$
for i= 1, 2, …
    solve $[M]z^{(i-1)}= r^{(i-1)}$
    $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
    if i=1
      $p^{(1)}= z^{(0)}$
    else
      $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
      $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
    endif
    $q^{(i)}= [A]p^{(i)}$
    $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
    $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
    $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
    check convergence |r|
end

solve_ICCG_mc (6/6)

```
!C
!C +----------------------+
!C | ALPHA= RHO / {p}{q}  |
!C +----------------------+
!C===
      C1= 0.d0
!$omp parallel do private(ip,i) reduction(+:C1)
      do ip= 1, PEsmpTOT
      do i = SMPindexG(ip-1)+1, SMPindexG(ip)
        C1= C1 + W(i,P)*W(i,Q)
      enddo
      enddo
!$omp end parallel do

      ALPHA= RHO / C1
!C===


!C
!C +----------------------+
!C | {x}= {x} + ALPHA*{p} |
!C | {r}= {r} - ALPHA*{q} |
!C +----------------------+
!C===
!$omp parallel do private(ip,i)
      do ip= 1, PEsmpTOT
      do i = SMPindexG(ip-1)+1, SMPindexG(ip)
        X(i)   = X(i)    + ALPHA * W(i,P)
        W(i,R)= W(i,R)  - ALPHA * W(i,Q)
      enddo
      enddo
!$omp end parallel do

      DNRM2= 0.d0
!$omp parallel do private(ip,i) reduction(+:DNRM2)
      do ip= 1, PEsmpTOT
      do i = SMPindexG(ip-1)+1, SMPindexG(ip)
        DNRM2= DNRM2 + W(i,R)**2
      enddo
      enddo
!$omp end parallel do
!C===
```

Compute $r^{(0)}= b-[A]x^{(0)}$
<u>for</u> i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  <u>if</u> i=1
    $p^{(1)}= z^{(0)}$
  <u>else</u>
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
  <u>endif</u>
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
  $\mathbf{x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}}$
  $\mathbf{r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}}$
  check convergence |r|
<u>end</u>

solve_ICCG_mc (6/6)

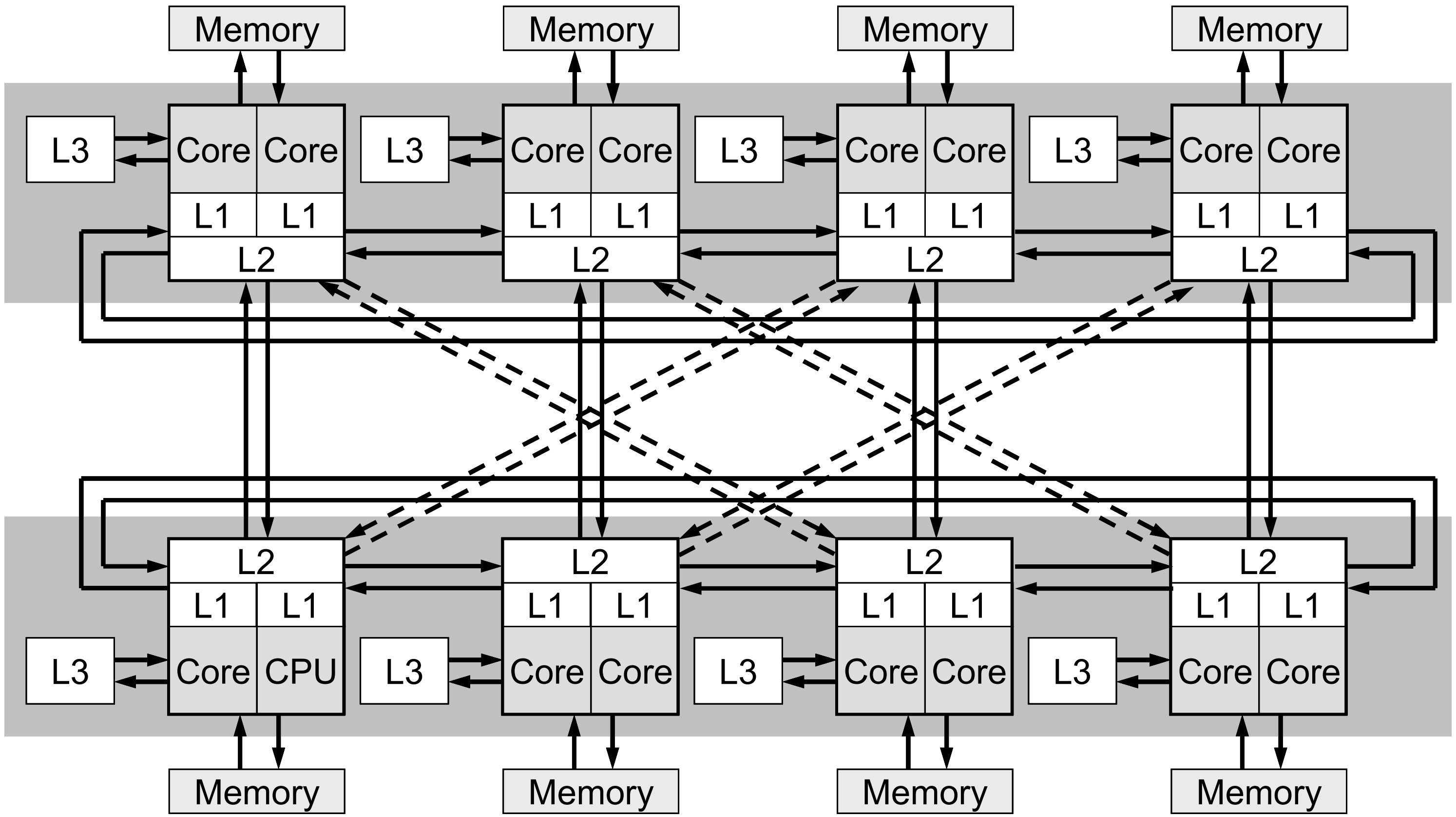
```
!C
!C +----------------------+
!C | ALPHA= RHO / {p}{q}  |
!C +----------------------+
!C===
      C1= 0.d0
!$omp parallel do private(ip,i) reduction(+:C1)
      do ip= 1, PEsmpTOT
      do i = SMPindexG(ip-1)+1, SMPindexG(ip)
        C1= C1 + W(i,P)*W(i,Q)
      enddo
      enddo
!$omp end parallel do

      ALPHA= RHO / C1
!C===


!C
!C +----------------------+
!C | {x}= {x} + ALPHA*{p} |
!C | {r}= {r} - ALPHA*{q} |
!C +----------------------+
!C===
!$omp parallel do private(ip,i)
      do ip= 1, PEsmpTOT
      do i = SMPindexG(ip-1)+1, SMPindexG(ip)
        X(i)  = X(i)   + ALPHA * W(i,P)
        W(i,R)= W(i,R) - ALPHA * W(i,Q)
      enddo
      enddo
!$omp end parallel do

      DNRM2= 0.d0
!$omp parallel do private(ip,i) reduction(+:DNRM2)
      do ip= 1, PEsmpTOT
      do i = SMPindexG(ip-1)+1, SMPindexG(ip)
        DNRM2= DNRM2 + W(i,R)**2
      enddo
      enddo
!$omp end parallel do
!C===
```

Compute $r^{(0)}= b-[A]x^{(0)}$
for i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
  endif
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence |r|
end

- Applying OpenMP to L2-sol
- **Examples**
- Optimization + Exercise

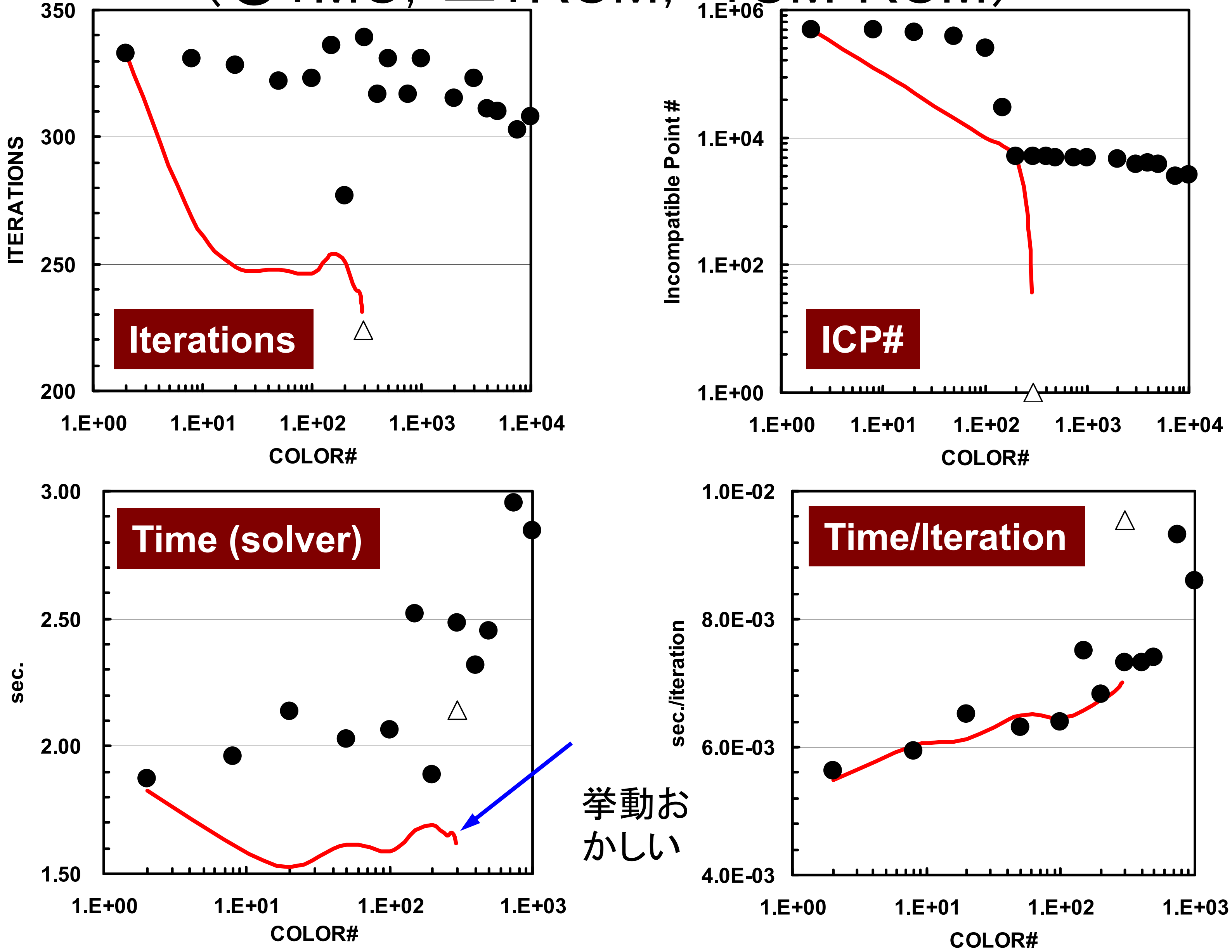
Results

- Hitachi SR11000/J2 1-node, 16-cores
- 100^3 Meshes



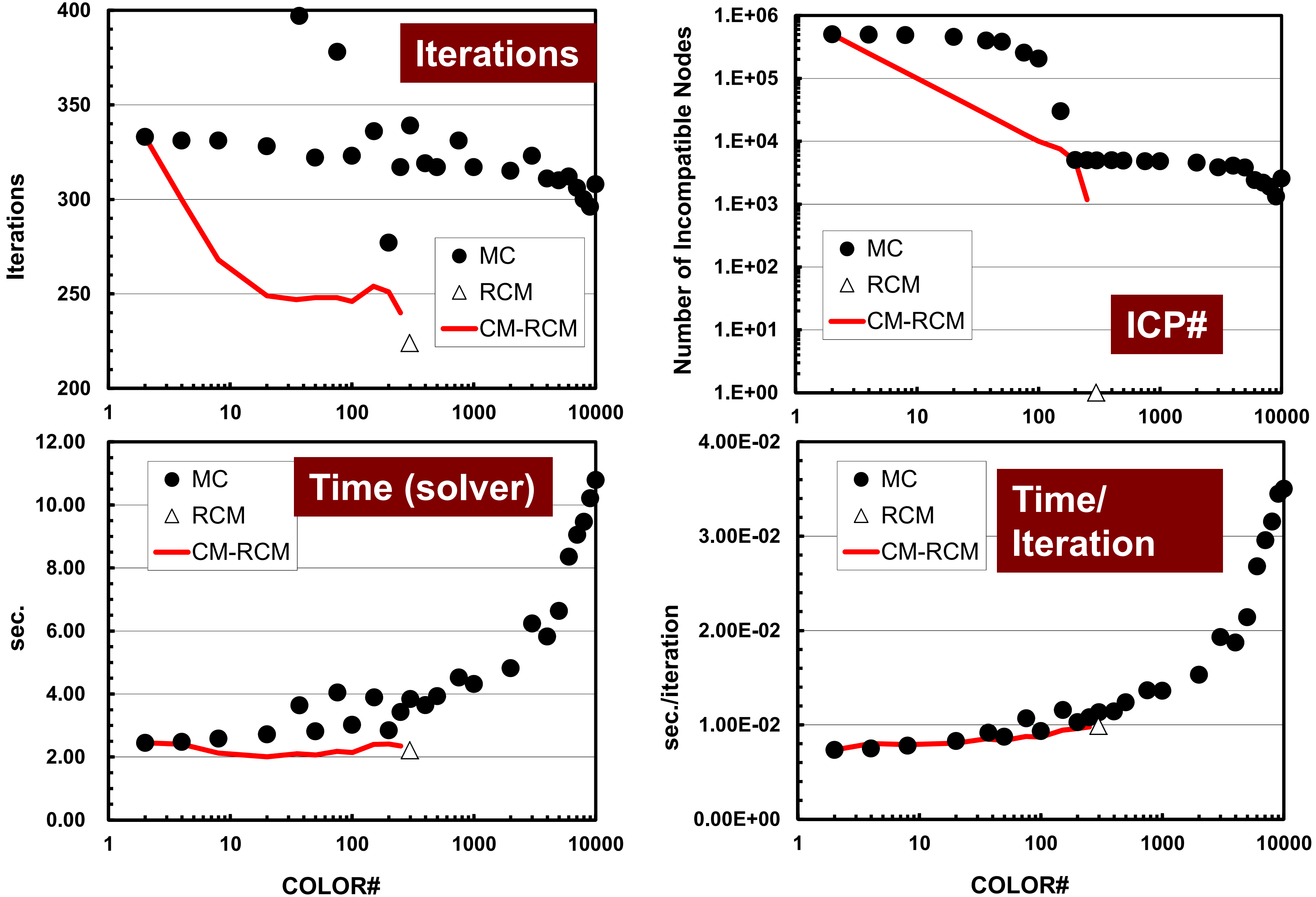
SR11000, 1-node/16-cores, 100^3

(●：MC, △：RCM, -：CM-RCM)



FX10, 1-node/16-cores, 100^3

(●：MC, △：RCM, -：CM-RCM)



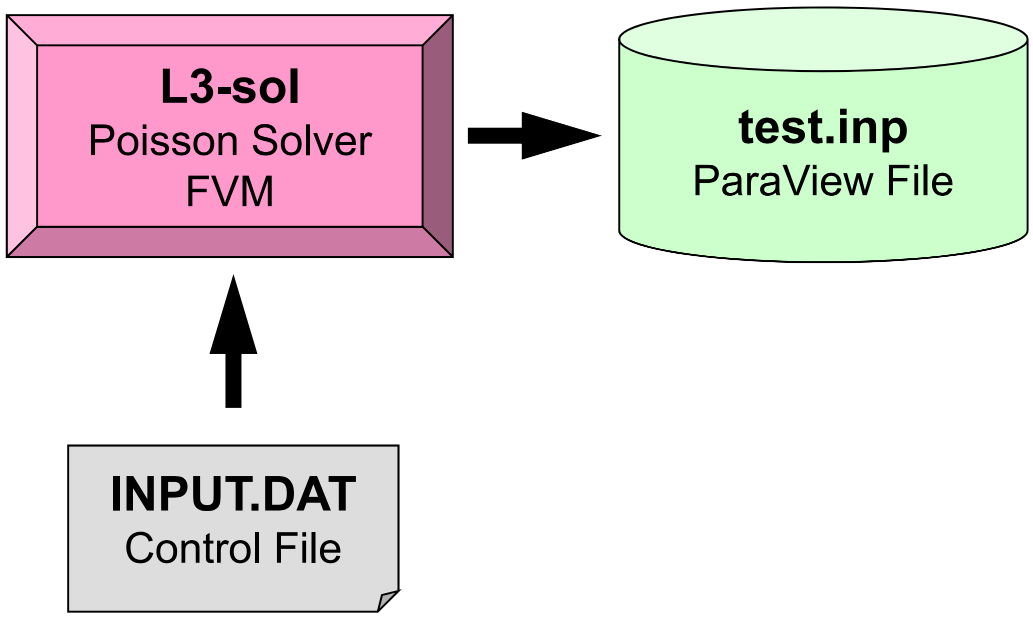
- Applying OpenMP to L2-sol
- Examples
- **Optimization + Exercise**

- Running the Code
- Further Optimization

Compile & Run

```
>$ cd <$O-L3>/src
>$ make
>$ ls ../run/L3-sol
  L3-sol
>$ cd ../run
>$ pjsub go1.sh
```

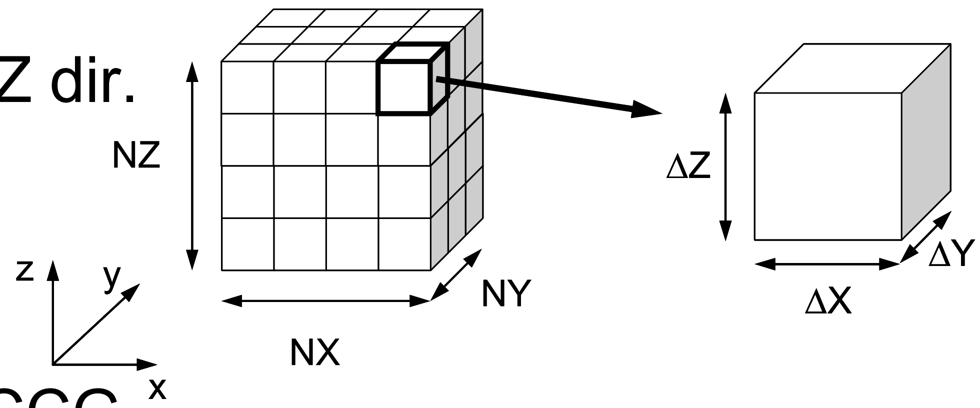
Running L3-sol



Control Data: INPUT.DAT

```
100 100 100                                  NX/NY/NZ
1.00e-00 1.00e-00 1.00e-00                   DX/DY/DZ
1.0e-08                                      EPSICCG
16                                           PEsmpTOT
100                                          NCOLORtot
```

- **NX,NY,NZ**
  – Number of meshes in X/Y/Z dir.
- **DX,DY,DZ**
  – Size of meshes
- **EPSICCG**
  – Convergence Criteria for ICCG
- **PEsmpTOT**
  – Thread Number **(= OMP_NUM_THREADS)**
- **NCOLORtot**
  – Reordering Method + Initial Number of Colors/Levels
  – ≧2: MC, =0: CM, =-1: RCM, -2≧: CMRCM

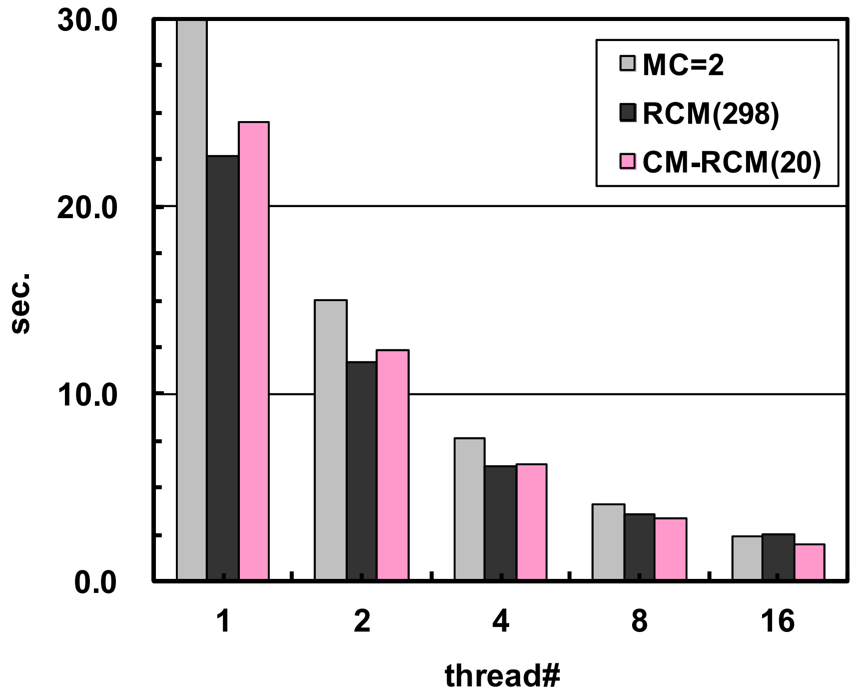


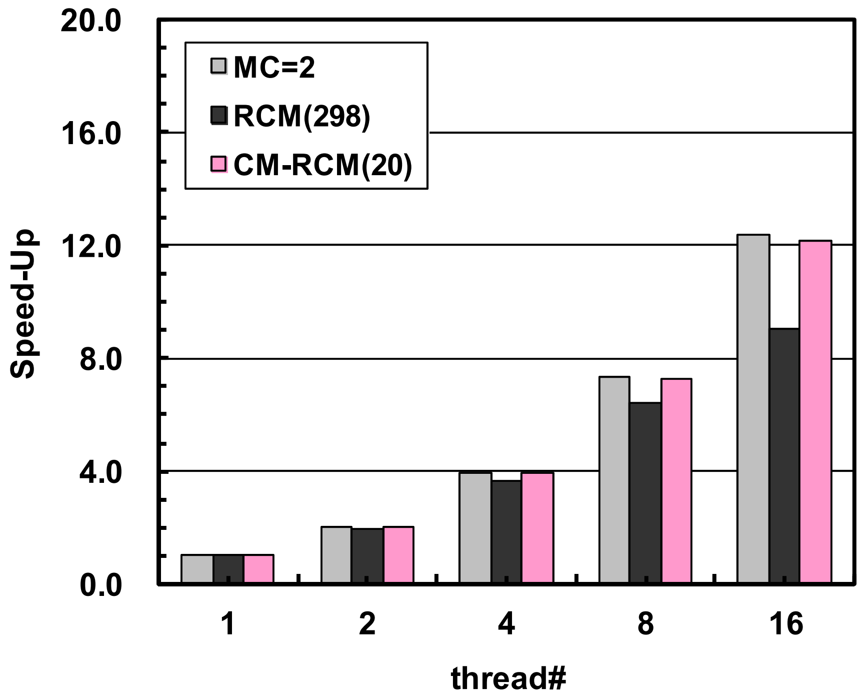
go1.sh

```
#!/bin/sh
#PJM -L "node=1"
#PJM -L "elapse=00:10:00"
#PJM -L "rscgrp=lecture9"
#PJM -g "gt99"
#PJM -j
#PJM -o “test.lst“

export OMP_NUM_THREADS=16     =PEsmpTOT
./L3-sol
```

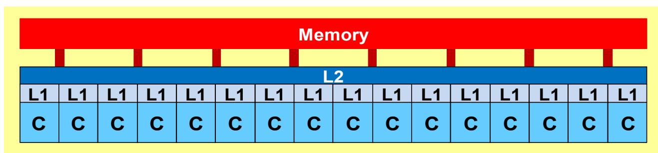
Results on FX10, 10^6 meshes

Iterations: MC(2): 333, RCM(298-levels): 224, CM-RCM(Nc=20): 249





16 threads
MC(2): 2.42 sec.
CM-RCM(20): 2.01 sec.



Exercise

- Various Configurations
  - Problem Size
  - Number of Threads
  - Number of Colors, Reordering Method (MC, RCM, CM-RCM)

- Running the Code
- **Further Optimization**
  - **OpenMP Statement**
  - Sequential Reordering
  - ELL

Forward Subst.: Current Impl. (F)

```
      do ic= 1, NCOLORtot
!$omp parallel do private(ip,ip1,i,WVAL,k)
        do ip= 1, PEsmpTOT
          ip1= (ic-1)*PEsmpTOT + ip
        do i= SMPindex(ip1-1)+1, SMPindex(ip1)
          WVAL= W(i,Z)
          do k= indexL(i-1)+1, indexL(i)
            WVAL= WVAL -  AL(k) * W(itemL(k),Z)
          enddo
          W(i,Z)= WVAL * W(i,DD)
        enddo
        enddo
!$omp end parallel do
      enddo
```

- At “`!omp parallel`”, generation and corruption of threads (up to 16) occurs.
  - In each color, this occurs
  - Some overhead
- Overhead increases, if number of color increases.

Forward Subst.: Current Impl. (C)

```
for(ic=0; ic<NCOLORtot; ic++) {
#pragma omp parallel for private (ip, ip1, i, WVAL, j)
        for(ip=0; ip<PEsmpTOT; ip++) {
                ip1 = ic * PEsmpTOT + ip;
                for(i=SMPindex[ip1]; i<SMPindex[ip1+1]; i++){
                        WVAL = W[Z][i];
                        for(j=indexL[i]; j<indexL[i+1]; j++){
                                WVAL -= AL[j] * W[Z][itemL[j]-1];
                        }
                        W[Z][i] = WVAL * W[DD][i];
                }
        }
}
```

- At “**`!omp parallel`**”, generation and corruption of threads (up to 16) occurs.
  - In each color, this occurs
  - Some overhead
- Overhead increases, if number of color increases.

For. Subst.: Reduced Overhead (F)

```
!$omp parallel private(ic,ip,ip1,i,WVAL,k)
      do ic= 1, NCOLORtot
!$omp do
        do ip= 1, PEsmpTOT
          ip1= (ic-1)*PEsmpTOT + ip
        do i= SMPindex(ip1-1)+1, SMPindex(ip1)
          WVAL= W(i,Z)
          do k= indexL(i-1)+1, indexL(i)
            WVAL= WVAL -  AL(k) * W(itemL(k),Z)
          enddo
          W(i,Z)= WVAL * W(i,DD)
        enddo
        enddo
      endd
!$omp end parallel
```

- Generation of threads occurs just once before starting forward substitutions.
- Loops with "`!omp do`" are parallelized.

For. Subst.: Reduced Overhead (C)

```
#pragma omp parallel private (ic, ip, ip1, i, WVAL, j)
for(ic=0; ic<NCOLORtot; ic++) {
#pragma omp for
        for(ip=0; ip<PEsmpTOT; ip++) {
                ip1 = ic * PEsmpTOT + ip;
                for(i=SMPindex[ip1]; i<SMPindex[ip1+1]; i++){
                        WVAL = W[Z][i];
                        for(j=indexL[i]; j<indexL[i+1]; j++){
                                WVAL -= AL[j] * W[Z][itemL[j]-1];
                        }
                        W[Z][i] = WVAL * W[DD][i];
                }
        }
}
```

- Generation of threads occurs just once before starting forward substitutions.
- Loops with "**`!omp do`**" are parallelized.

Programs

```
>$ cd <$O-L3>/src0
>$ make
>$ ls ../run/L3-sol0
  L3-sol0
>$ cd ../run
(modify "go0.sh")
>$ pjsub go0.sh
```

Results: L3-sol0 is better $N=128^3$ (Fortran, W(N,4), and so on …)

	L3-sol	L3-sol0
NCOLORtot= -20 CM-RCM (20) 318 Iterations	5.69 sec.	5.44 sec.
NCOLORtot= -1 RCM (382 levels) 287 Iterations	6.54 sec.	6.37 sec.

- Running the Code
- **Further Optimization**
  - OpenMP Statement
  - **Sequential Reordering**
  - **ELL**

Problems in Reordering

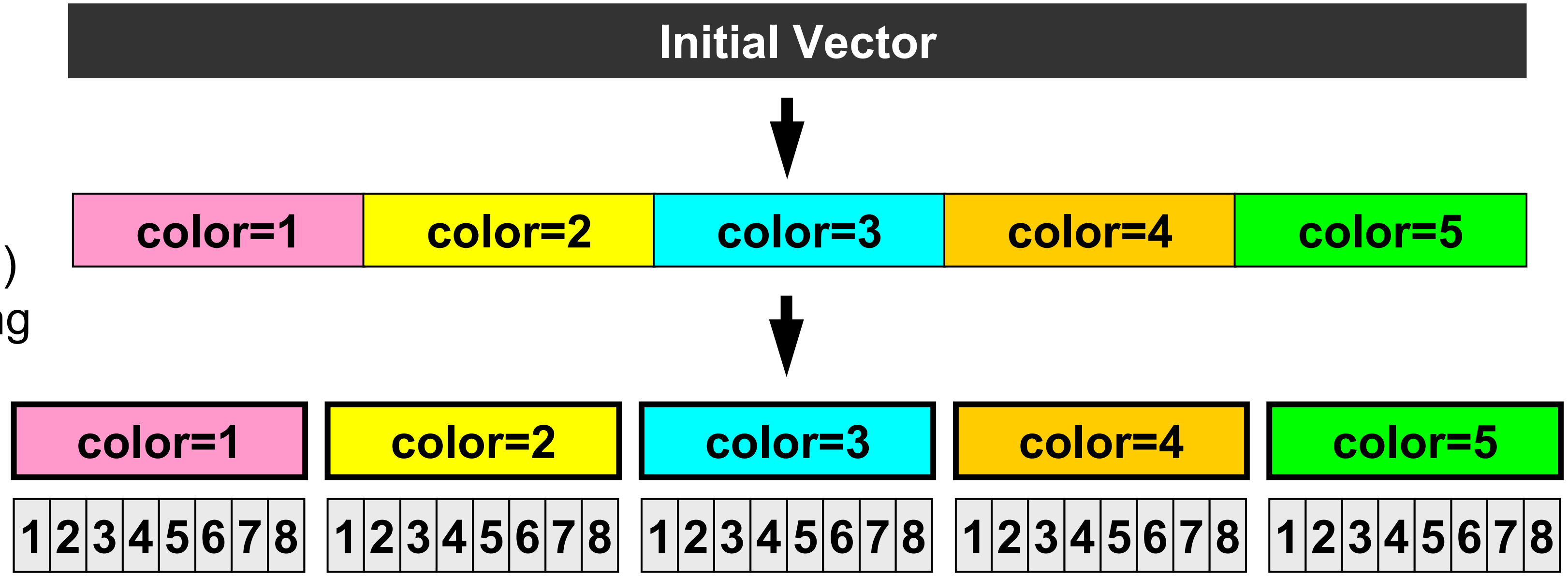
- Coloring
  - MC
  - RCM
  - CM-RCM
- Renumbering is according to color/level ID
- On each thread, numbering is not continuous
  - reduced performance

SMPindex:

for preconditioning

```
  do ic= 1, NCOLORtot
!$omp parallel do …
    do ip= 1, PEsmpTOT
      ip1= (ic-1)*PEsmpTOT+ip
    do i= SMPindex(ip1-1)+1, SMPindex(ip1)
      (…)
    enddo
    enddo
!omp end parallel do
  enddo
```

Coloring
(5 colors)
+Ordering



- 5-colors, 8-threads
- Meshes in same color are independent: parallel processing
- Reordering in ascending order according to color ID

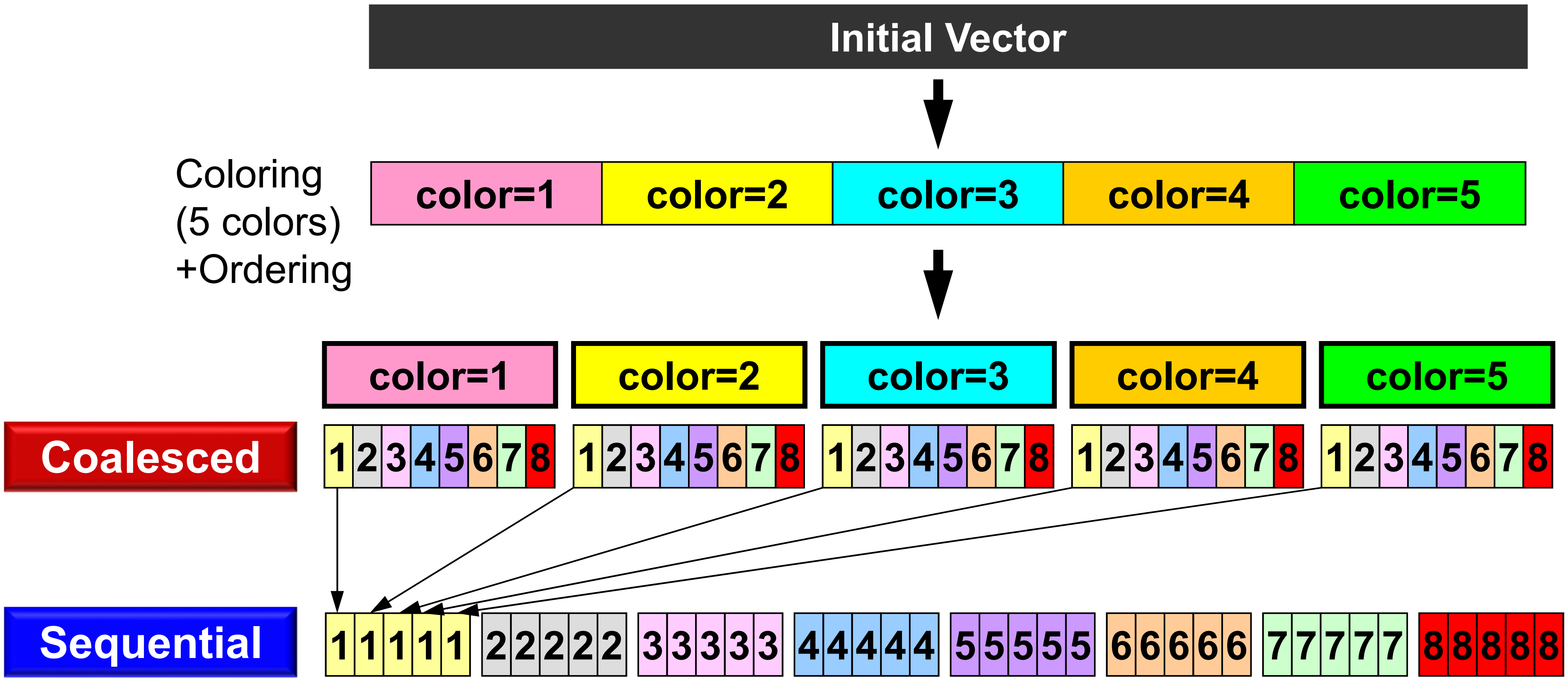
Sequential Reordering

- Reordering for continuous memory access on each thread (core)
  - Performance is expected to be better.
    - Continuous address of arrays, such as coefficient matrices
    - Locality (2-page later)

- Inconsistent numbering
  - `itemL(k) > icel`
  - `indexL(icel-1)+1≦k≦indexL(icel)`

Sequential Reordering

Further reordering for continuous memory access on each thread, 5-color, 8-threads



Sequential Reordering

CM-RCM(2), 4-threads

Continuous Data Access on a Thread: Utilization of Cache, Prefetching

45	10	39	5	35	2	33	1
17	46	11	40	6	36	3	34
53	18	47	12	41	7	37	4
24	54	19	48	13	42	8	38
59	25	55	20	49	14	43	9
29	60	26	56	21	50	15	44
63	30	61	27	57	22	51	16
32	64	31	62	28	58	23	52

CM-RCM(2)

29	18	15	5	11	2	9	1
33	30	19	16	6	12	3	10
45	34	31	20	25	7	13	4
40	46	35	32	21	26	8	14
59	49	47	36	41	22	27	17
53	60	50	48	37	42	23	28
63	54	61	51	57	38	43	24
56	64	55	62	52	58	39	44

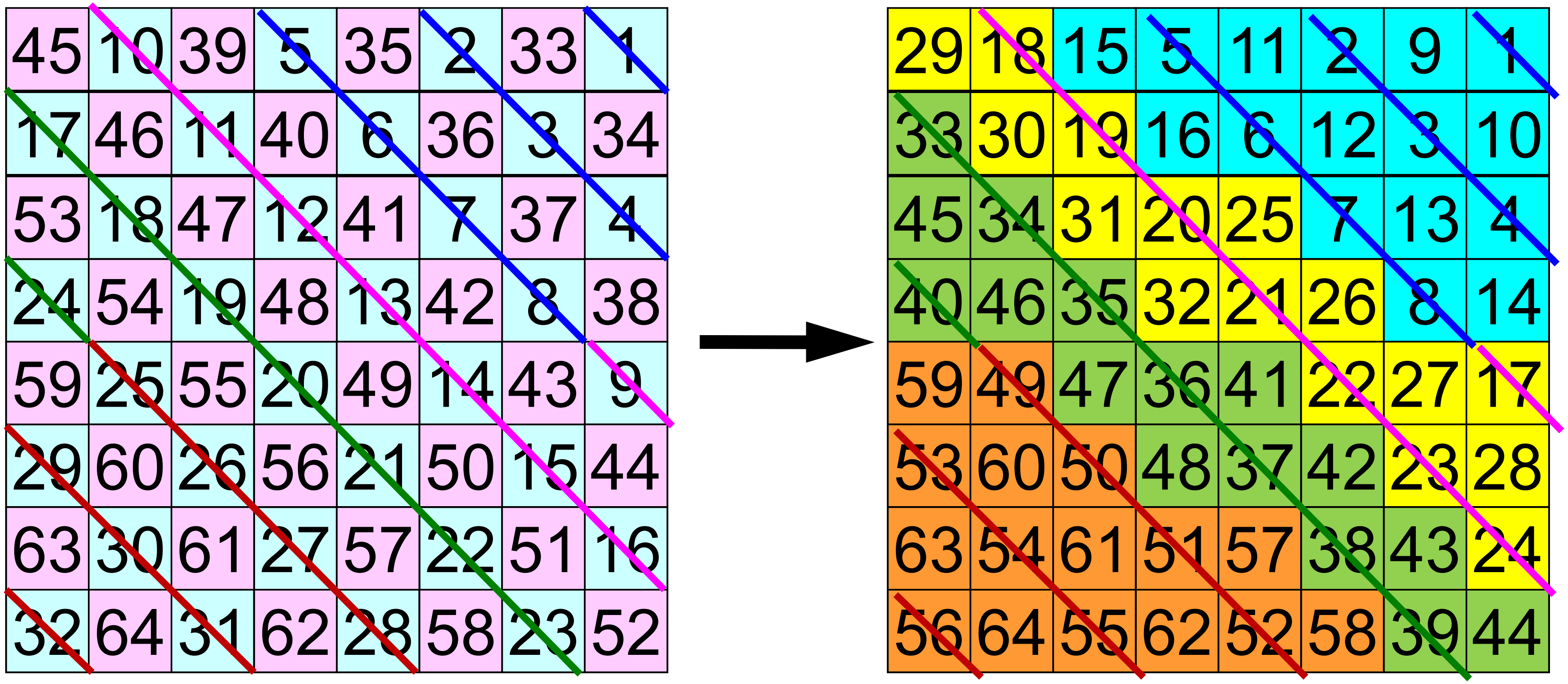
Sequential Reordering, 4-threads

Sequential Reordering

CM-RCM(2), 4-threads

1st-Color

#0 thread, #1, #2, #3



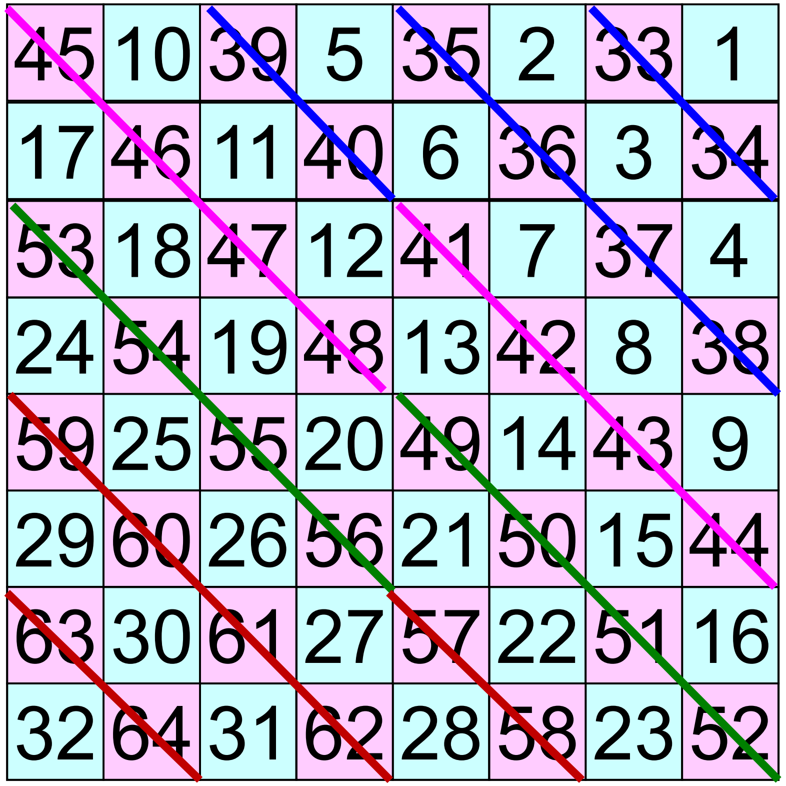
CM-RCM(2) **Sequential Reordering, 4-threads**

Sequential Reordering

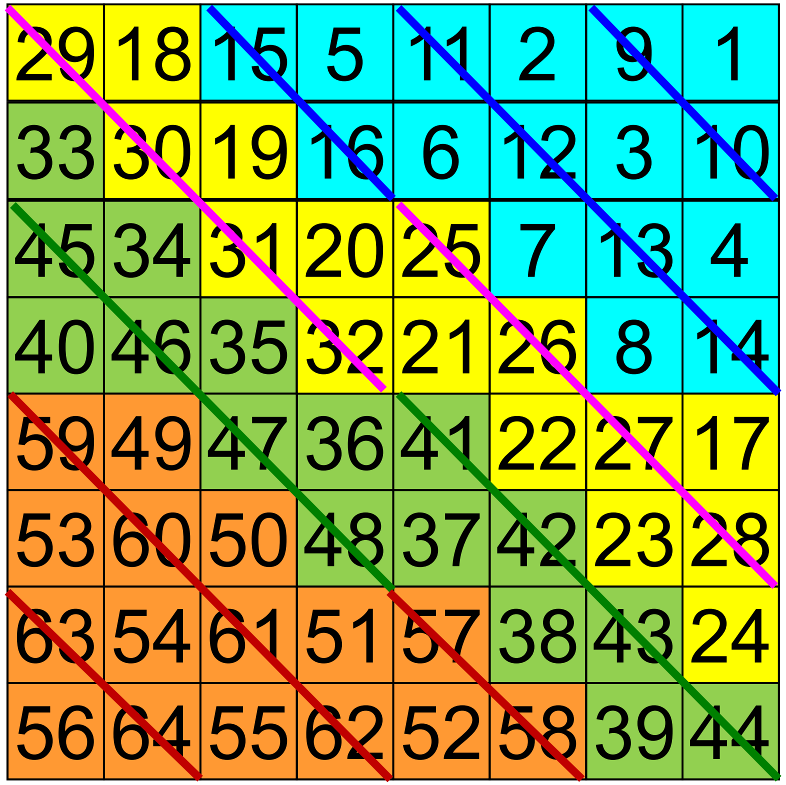
CM-RCM(2), 4-threads

2nd-Color

#0 thread, #1, #2, #3



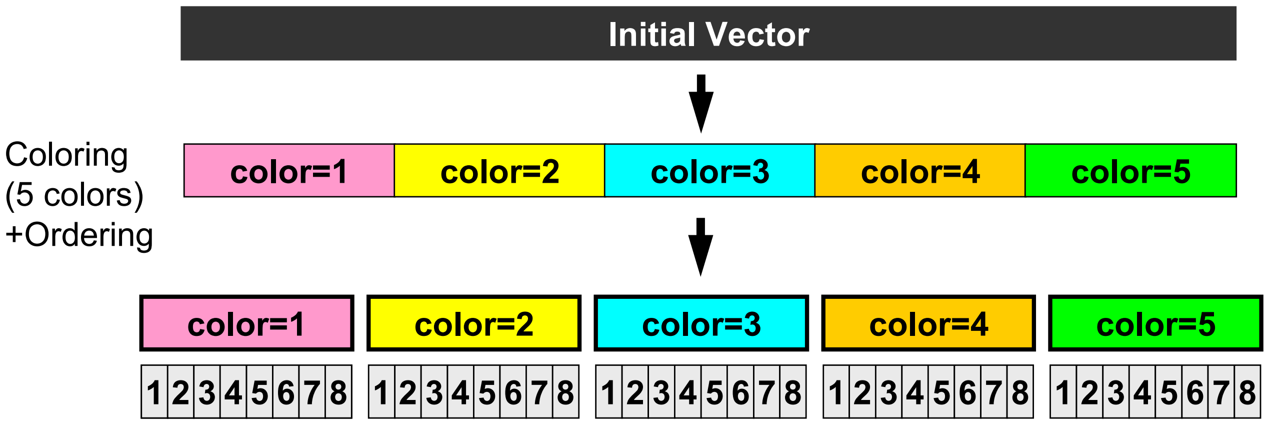
CM-RCM(2)



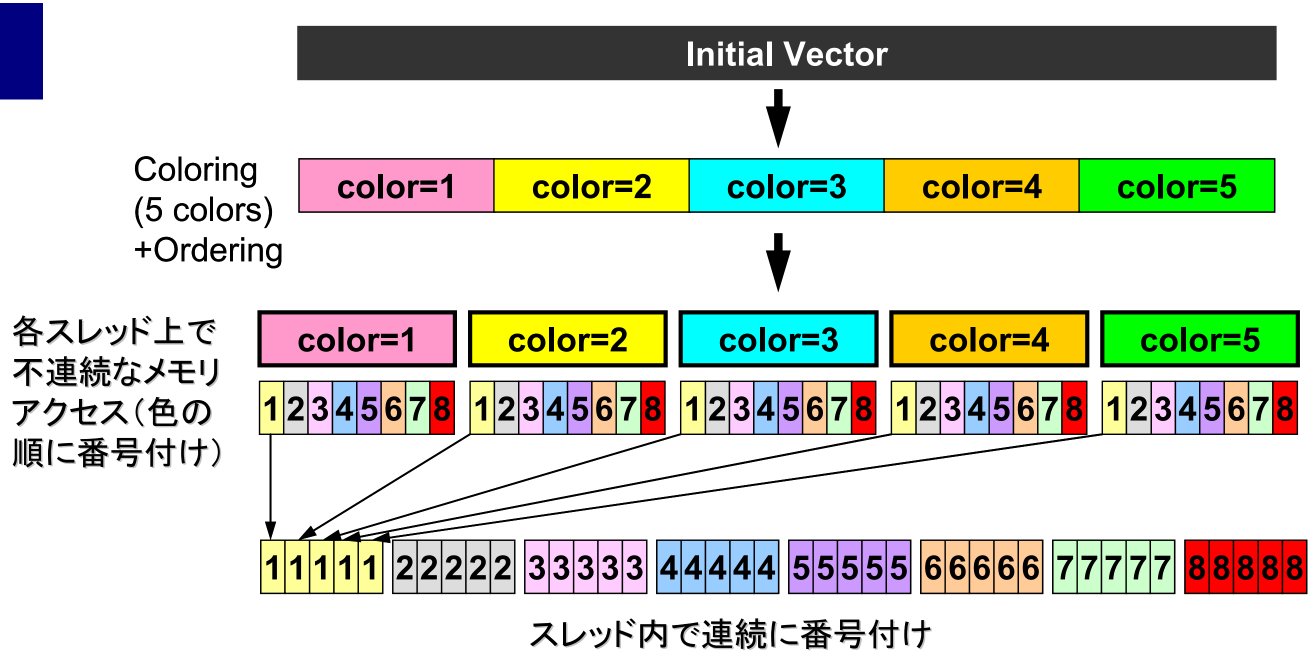
Sequential Reordering, 4-threads

Sequential Reordering

Coalesced
Good for GPU



Sequential



Files

```
>$ cd <$O-L3>/reorder0
>$ make
>$ ls ../run/L3-rsol0
  L3-rsol0
>$ cd ../run
(modify “gor.sh”)
>$ pjsub gor.sh
```

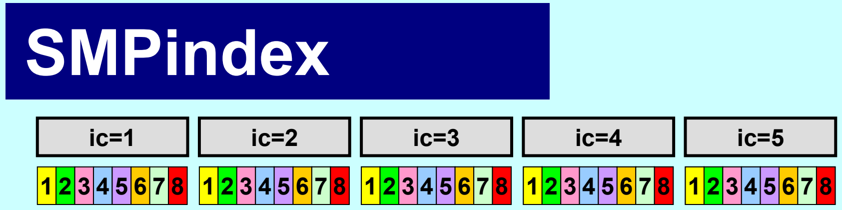
INPUT.DAT

```
100 100 100                        NX/NY/NZ
1.00e-00 1.00e-00 1.00e-00         DX/DY/DZ
1.0e-08                            EPSICC
16                                 PEsmpTOT
100                                NCOLORtot
0                                  NFLAG
0                                  METHOD
```

- **PEsmpTOT**
  - Thread Number **(= OMP_NUM_THREADS)**
- **NCOLORtot**
  - Reordering Method + Initial Number of Colors/Levels
  - ≧2: MC, =0: CM, =-1: RCM, -2≧: CMRCM
- **NFLAG**
  - =0: without first-touch, =1: with first-touch
- **METHOD**
  - Loop structure for Mat-Vec
  - =0: conventional way, =1: similar to forward/backward substitution

Sequential Reordering

```
allocate (SMPindex(0:PEsmpTOT*NCOLORtot))
SMPindex= 0
do ic= 1, NCOLORtot
  nn1= COLORindex(ic) - COLORindex(ic-1)
  num= nn1 / PEsmpTOT
  nr = nn1 - PEsmpTOT*num
  do ip= 1, PEsmpTOT
    if (ip.le.nr) then
      SMPindex((ic-1)*PEsmpTOT+ip)= num + 1
     else
      SMPindex((ic-1)*PEsmpTOT+ip)= num
    endif
  enddo
enddo

allocate (SMPindex_new(0:PEsmpTOT*NCOLORtot))
SMPindex_new(0)= 0
do ic= 1, NCOLORtot
  do ip= 1, PEsmpTOT
    j1= (ic-1)*PEsmpTOT + ip
    j0= j1 - 1
    SMPindex_new((ip-1)*NCOLORtot+ic)= SMPindex(j1)
    SMPindex(j1)= SMPindex(j0) + SMPindex(j1)
  enddo
enddo

do ip= 1, PEsmpTOT
  do ic= 1, NCOLORtot
    j1= (ip-1)*NCOLORtot + ic
   j0= j1 - 1
    SMPindex_new(j1)= SMPindex_new(j0) + SMPindex_new(j1)
  enddo
enddo
```



Mat-Vec

```
!$omp parallel do private(ip,i,VAL,k)
      do ip= 1, PEsmpTOT
        do i = SMPindexG(ip-1)+1, SMPindexG(ip)
          VAL= D(i)*W(i,P)
          do k= indexL(i-1)+1, indexL(i)
            VAL= VAL + AL(k)*W(itemL(k),P)
          enddo
          do k= indexU(i-1)+1, indexU(i)
            VAL= VAL + AU(k)*W(itemU(k),P)
          enddo
          W(i,Q)= VAL
        enddo
      enddo
!$omp end parallel do
```

METHOD=0

```
!$omp parallel do private(ip,i,VAL,k)
      do ip= 1, PEsmpTOT
        do i= SMPindex((ip-1)*NCOLORtot)+1, SMPindex(ip*NCOLORtot)
          VAL= D(i)*W(i,P)
          do k= indexL(i-1)+1, indexL(i)
            VAL= VAL + AL(k)*W(itemL(k),P)
          enddo
          do k= indexU(i-1)+1, indexU(i)
            VAL= VAL + AU(k)*W(itemU(k),P)
          enddo
          W(i,Q)= VAL
        enddo
      enddo
!$omp end parallel do
```

METHOD=1

Forward Substitution

```
!$omp parallel private(ip,ip1,i,WVAL,k)
      do ic= 1, NCOLORtot
!$omp do
        do ip= 1, PEsmpTOT
          ip1= (ic-1)*PEsmpTOT + ip
        do i= SMPindex(ip1-1)+1, SMPindex(ip1)
          WVAL= W(i,Z)
          do k= indexL(i-1)+1, indexL(i)
            WVAL= WVAL -  AL(k) * W(itemL(k),Z)
          enddo
          W(i,Z)= WVAL * W(i,DD)
        enddo
        enddo
      enddo
!$omp end parallel
```

Original

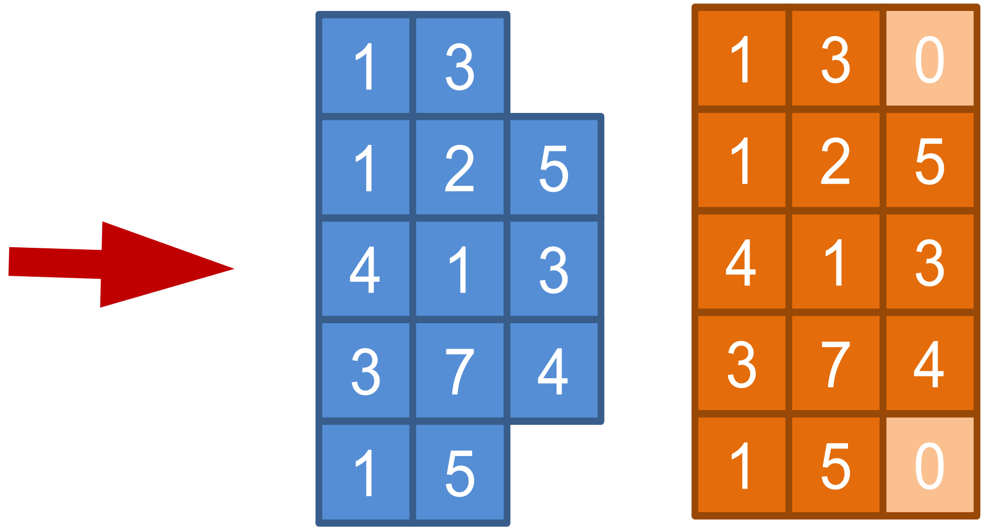
```
!$omp parallel private(ip,ip1,i,WVAL,k)
      do ic= 1, NCOLORtot
!$omp do
        do ip= 1, PEsmpTOT
          ip1= (ip-1)*NCOLORtot + ic
        do i= SMPindex(ip1-1)+1, SMPindex(ip1)
          WVAL= W(i,Z)
          do k= indexL(i-1)+1, indexL(i)
            WVAL= WVAL -  AL(k) * W(itemL(k),Z)
          enddo
          W(i,Z)= WVAL * W(i,DD)
        enddo
        enddo
      enddo
!$omp end parallel
```

New

Matrix Storage Format

ELL (Ellpack-Itpack): Fixed Loop Length, Good for Prefetching

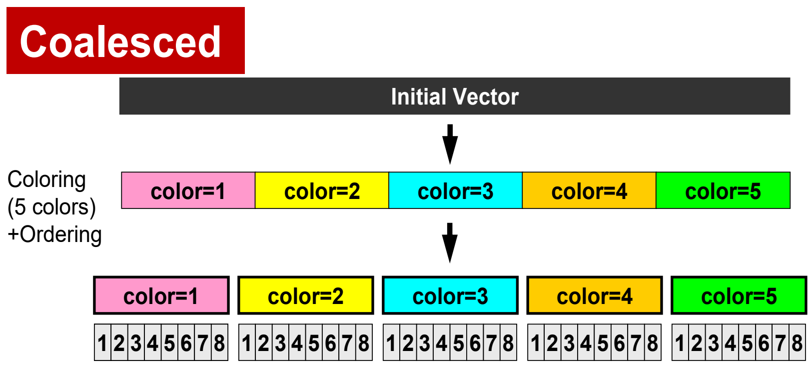
$$\begin{bmatrix} 1 & 3 & 0 & 0 & 0 \\ 1 & 2 & 5 & 0 & 0 \\ 4 & 1 & 3 & 0 & 0 \\ 0 & 3 & 7 & 4 & 0 \\ 1 & 0 & 0 & 0 & 5 \end{bmatrix}$$

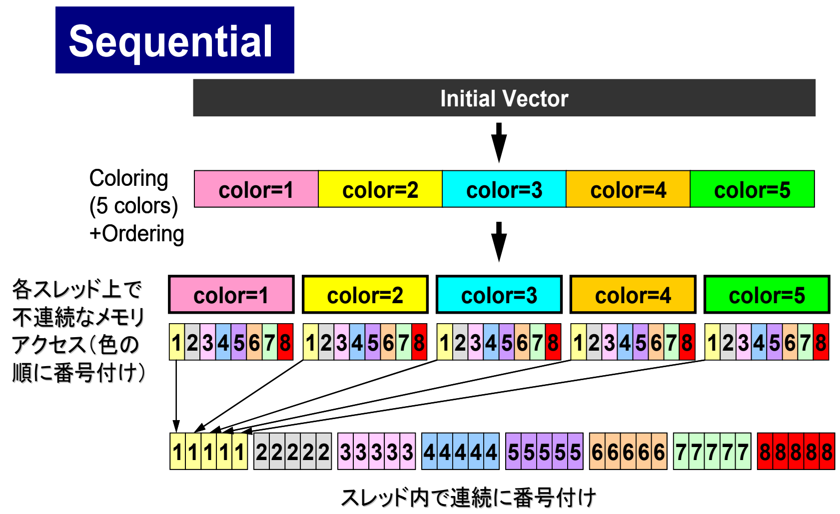


(a) CRS (b) ELL

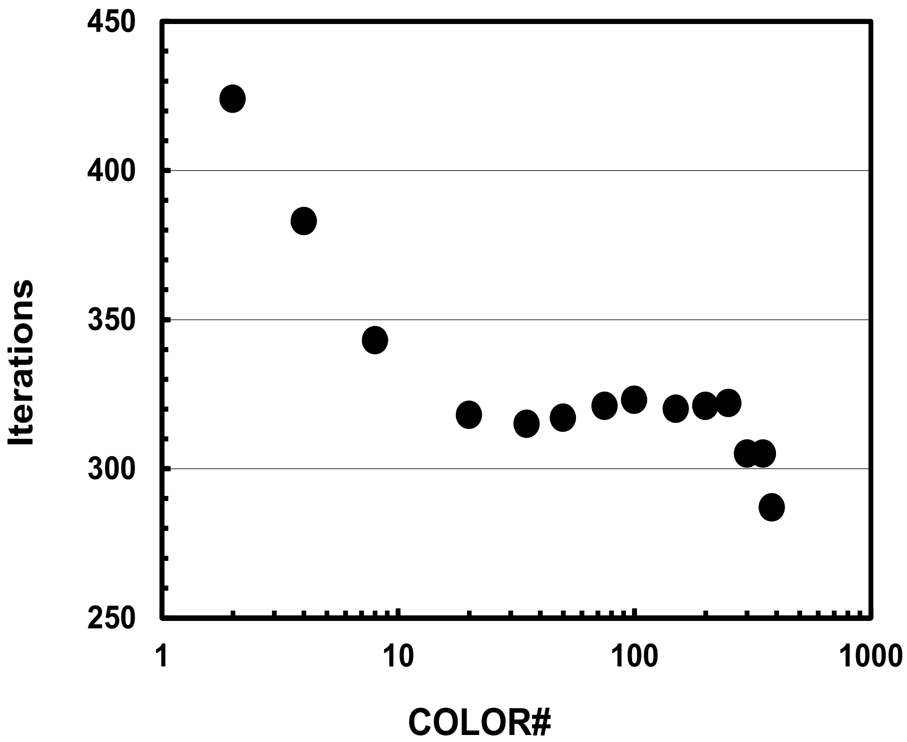
Cases: 128^3 meshes

		Coloring	Further Reordering	First Touch Data Placement	Matrix Storage Format
src0	Case-1	CM-RCM	Coalesced (図 4 (a))	NO	CRS
reorder0	Case-2	CM-RCM	Sequential (図 4 (b))	YES	CRS
ELL	Case-3	CM-RCM	Sequential (図 4 (b))	YES	ELL



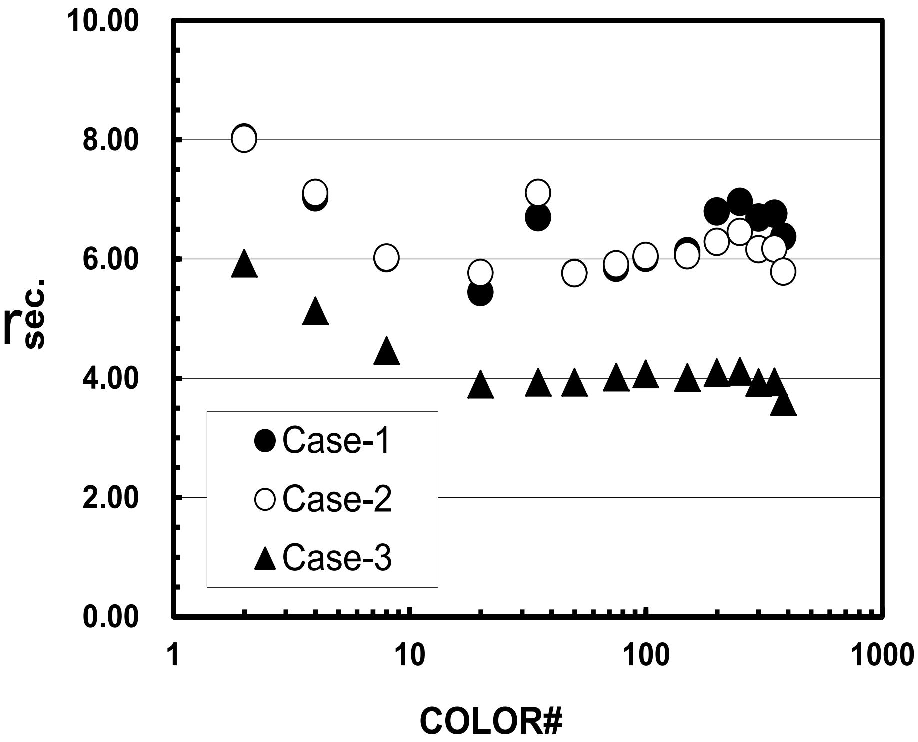


Color# ~ Iteration
CM-RCM



Results: FX10

- CASE-1(src0)⇒CASE-2(reorder0)
  - Slightly improved when number of colors are larger
  - Generally speaking, performance is getting worse if number of colors increases
    - In CASE-2, data on each thread is continuous, when computation proceeds to the next color.
  - First Touch: NO effect
- ELL: Big effect

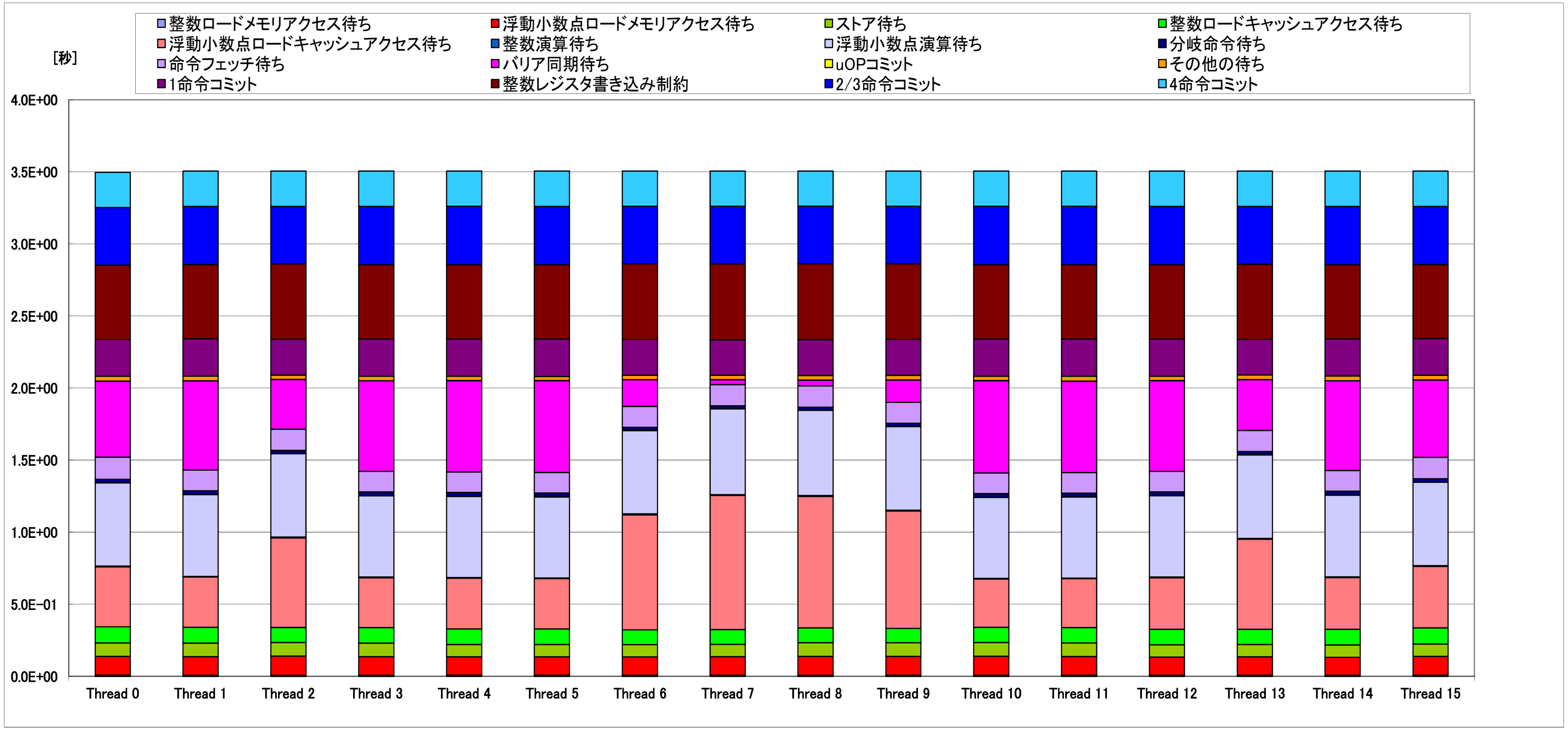


Case-1: src0
Case-2: reorder0
Case-3: reorder0 + ELL

Fujitsu FX10: CASE-1, CM-RCM(2)

L1-dem.-miss:25.6%, Mem. throughput:41.8GB/sec.

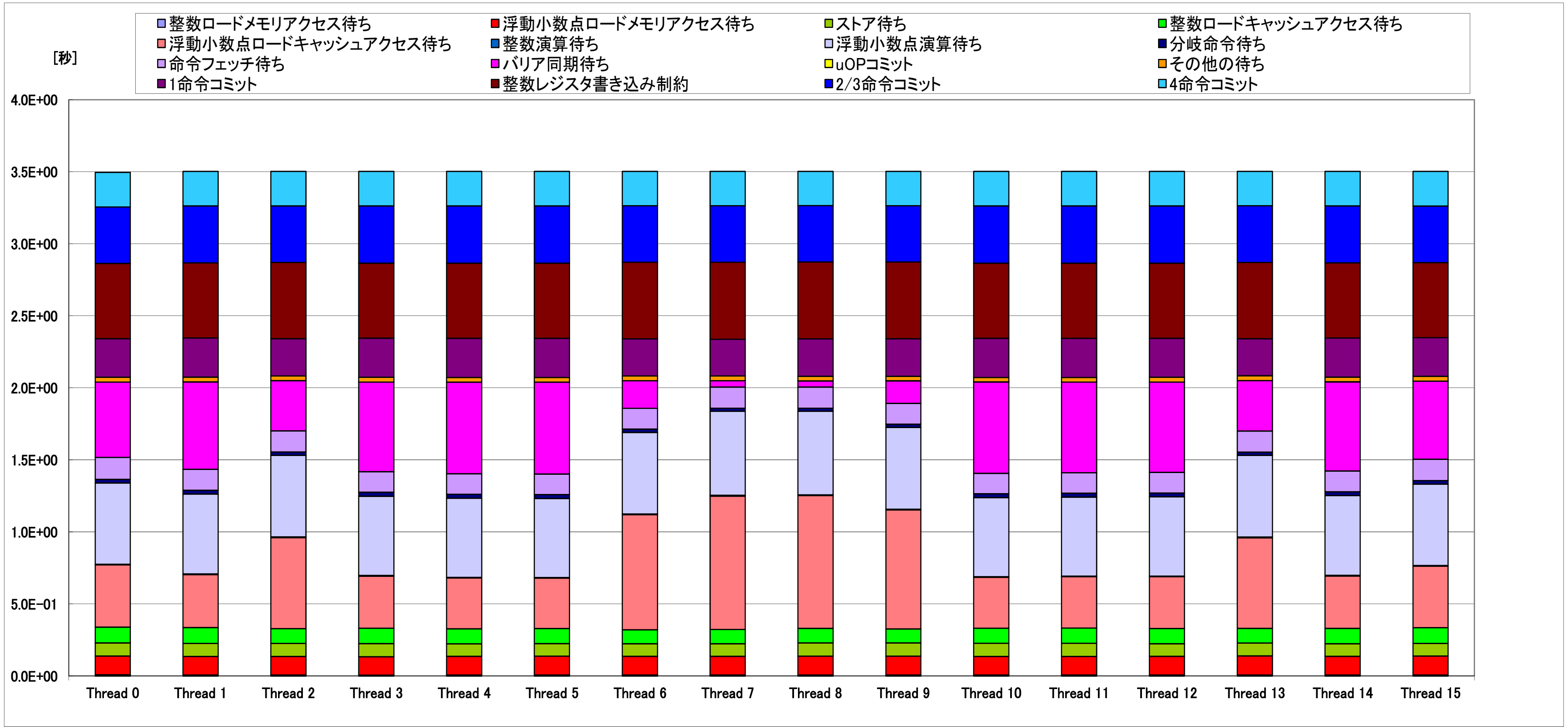
Forward/Backward Substitution



src0: CRS, Coalesced

Fujitsu FX10: CASE-2, CM-RCM(2)

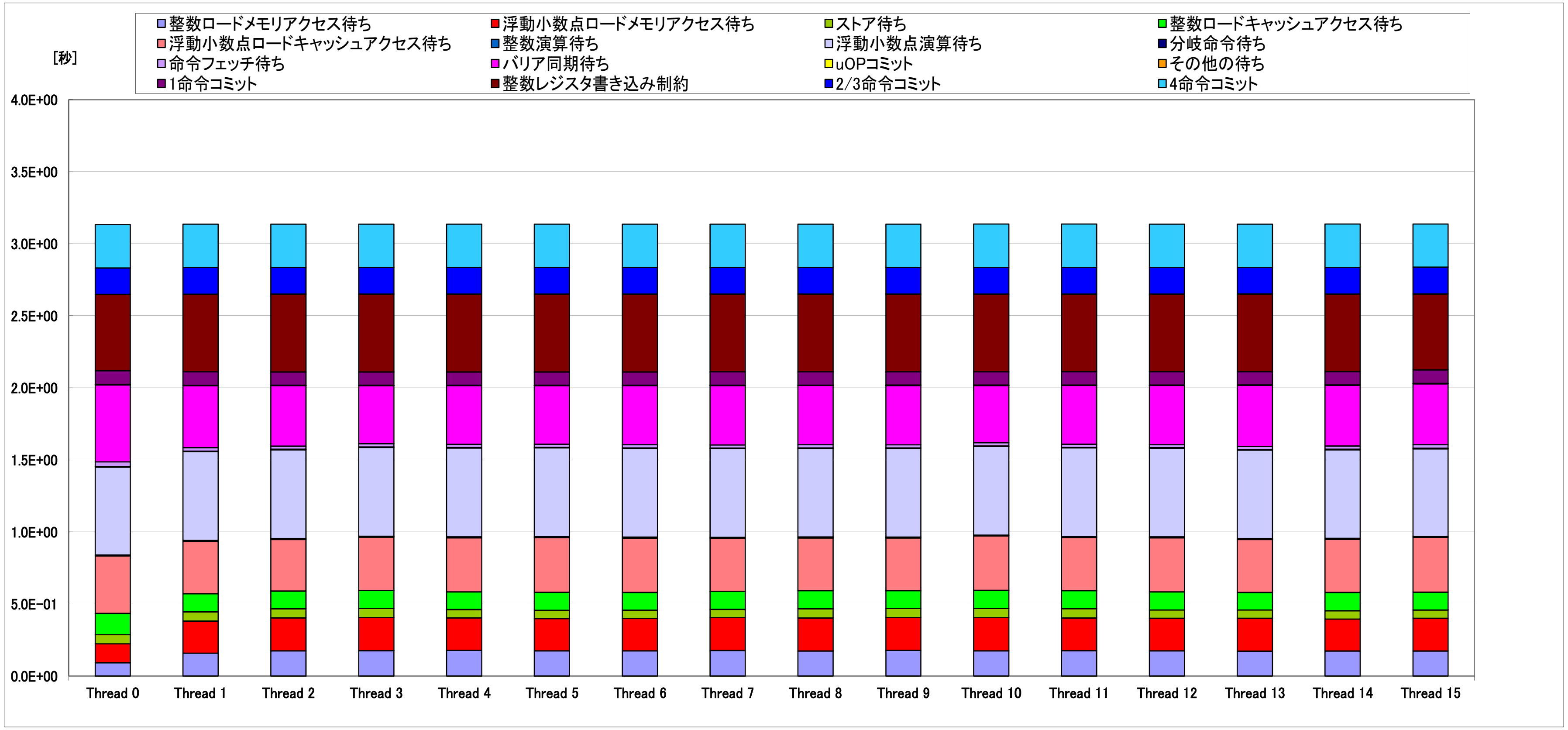
25.6%, 41.8GB/sec.



reorder0: CRS, Sequential

Fujitsu FX10: CASE-1, CM-RCM(382)

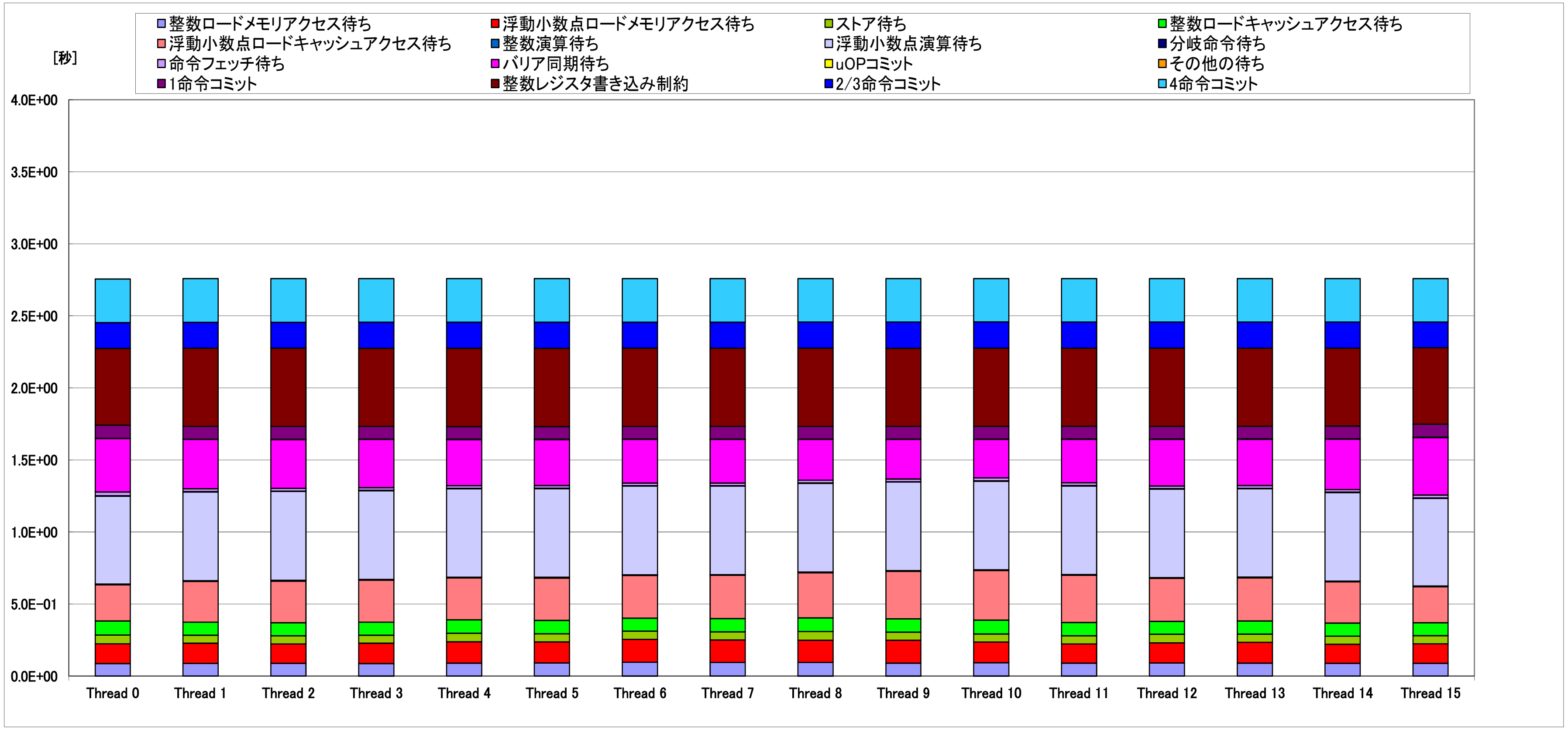
37.7%, 28.7GB/sec.



src0: CRS, Coalesced

Fujitsu FX10: CASE-2, CM-RCM(382)

29.3%, 32.6GB/sec.



reorder0: CRS, Sequential

Summary: Fujitsu FX10 Analysis by Profiler

Upper: L1 Demand Miss Rate
Lower: Memory Throughput

	src0 CASE-1 CRS+ Coalesced	reorder0 CASE-2 CRS+ Sequential	CASE-3 ELL+ Sequential
CM-RCM(2)	25.5 %	25.6 %	5.42 %
	41.8 GB/sec.	41.8 GB/sec.	64.0 GB/sec.
CM-RCM(382)	37.7 %	29.3 %	16.5 %
	28.7 GB/sec.	32.6 GB/sec.	52.2 GB/sec.

Summary: Fujitsu FX10 Analysis by Profiler

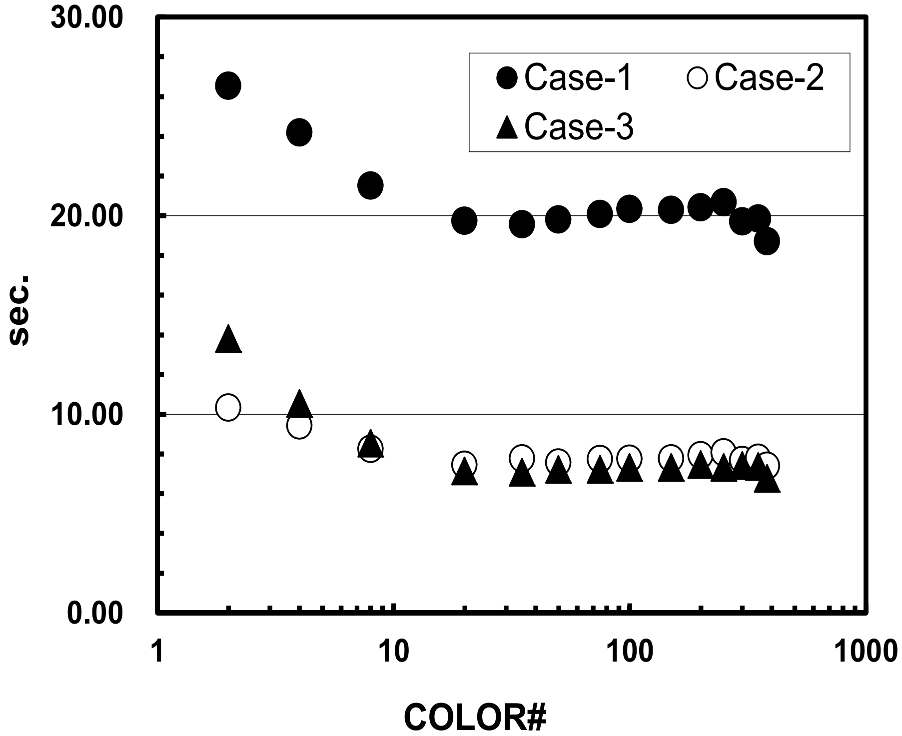
Upper: CM-RCM(20), Lower: CM-RCM(382)

	Instructions	SIMD（%）	Memory Access Throughput（%）
Case-2 CRS	1.83×10^{11} 1.83×10^{11}	7.17 6.90	50.2 44.5
Case-3 ELL	6.71×10^{10} 5.96×10^{10}	16.3 16.2	69.8 67.0

Case-1: src0
Case-2: reorder0
Case-3: reorder0 + ELL

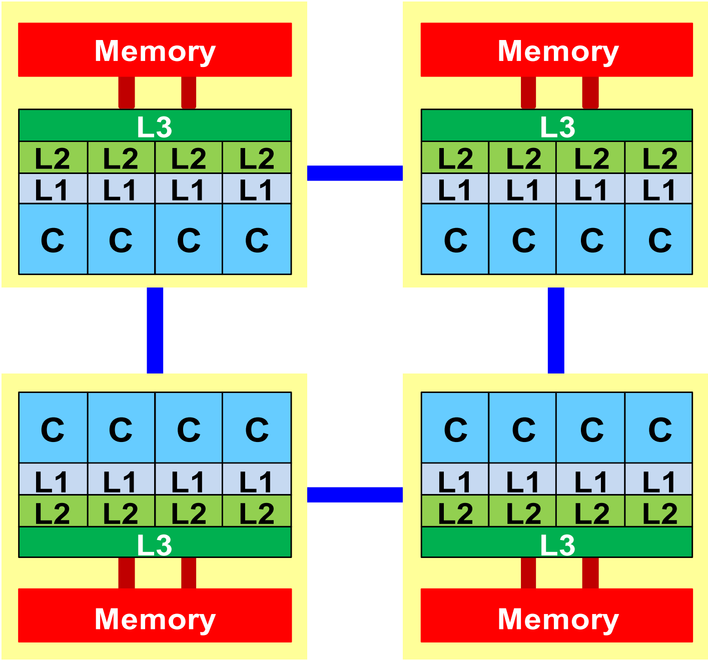
Results: Cray XE6

- CASE-1(src0)⇒ CASE-2(reorder0)
  - Significant Improvement
  - Optimization for NUMA Architecture
  - + First Touch
- CRS⇒ELL Improvement is not so large

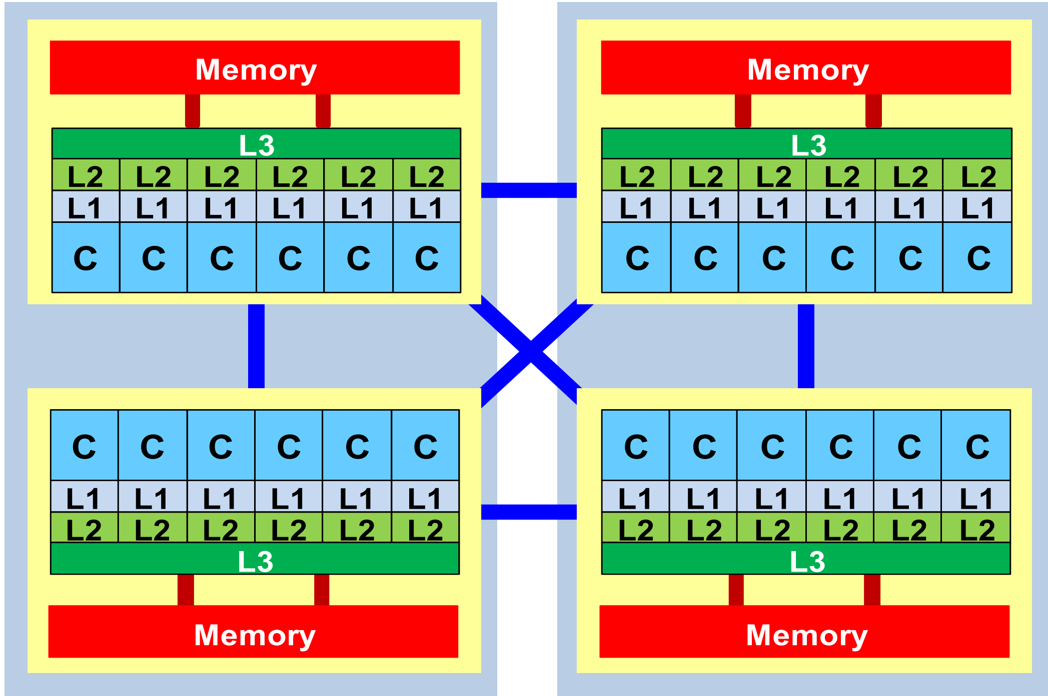


Case-1: src0
Case-2: reorder0
Case-3: reorder0 + ELL

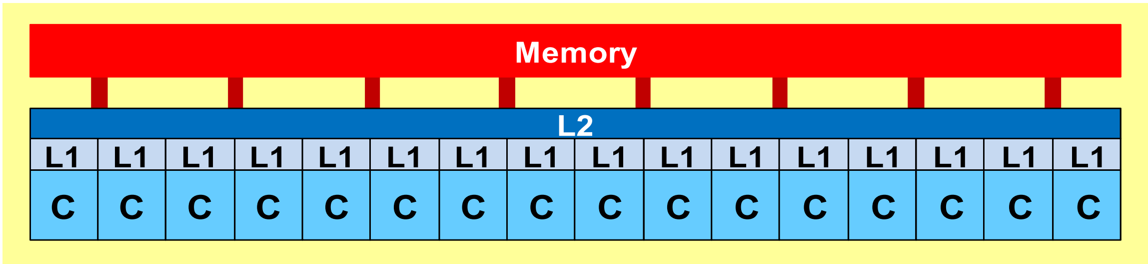
T2K/Tokyo



Cray XE6 (Hopper)



Fujitsu FX10 (Oakleaf-FX)



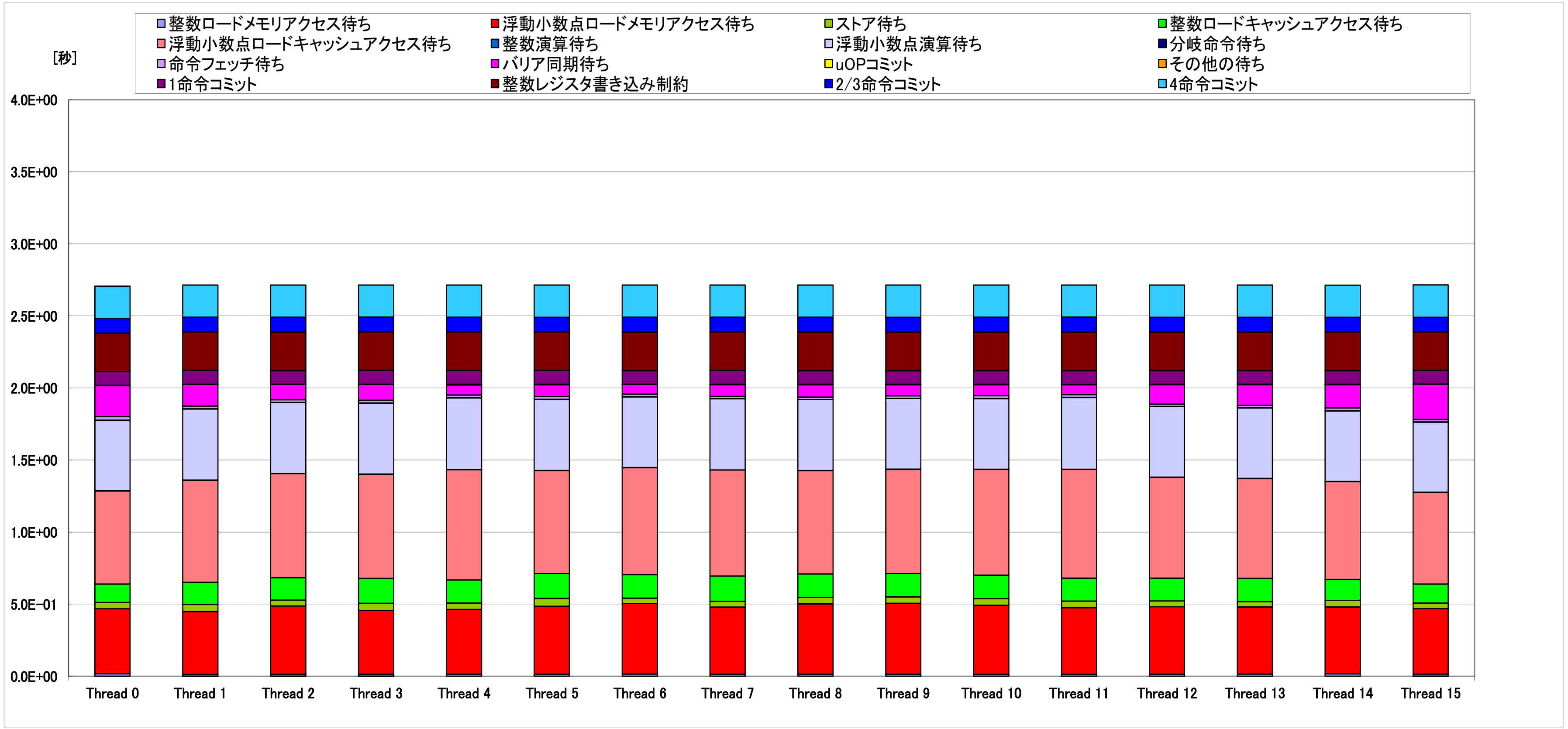
Summary

		CM-RCM(20)		CM-RCM(382) = RCM	
		Time (sec.)	Time/Iteration (sec.)	Time (sec.)	Time/Iteration (sec.)
Fujitsu FX10	Case-1	5.44	1.71×10^{-2}	6.37	2.22×10^{-2}
	Case-2	5.76	1.81×10^{-2}	5.78	2.02×10^{-2}
	Case-3	3.90	1.23×10^{-2}	3.61	1.26×10^{-2}
Cray XE6	Case-1	19.7	6.26×10^{-2}	18.7	6.52×10^{-2}
	Case-2	7.45	2.34×10^{-2}	7.40	2.58×10^{-2}
	Case-3	7.14	2.25×10^{-2}	6.77	2.36×10^{-2}

Case-1: src0
Case-2: reorder0
Case-3: reorder0 + ELL

Fujitsu FX10: CASE-3, CM-RCM(2)

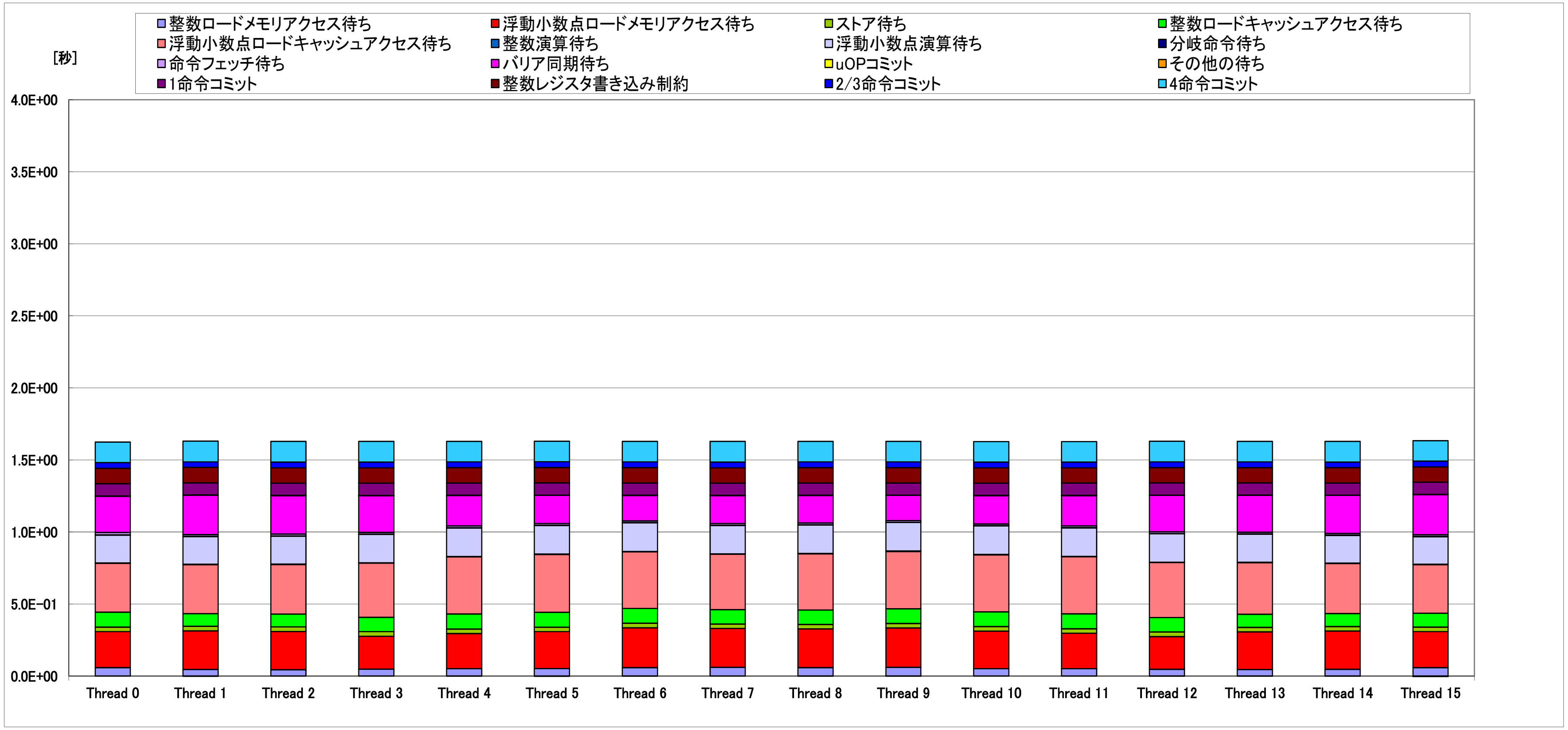
5.4%, 64.0GB/sec.



ELL, Sequential

Fujitsu FX10: CASE-3, CM-RCM(382)

16.5%, 52.2GB/sec.



ELL, Sequential

Summary

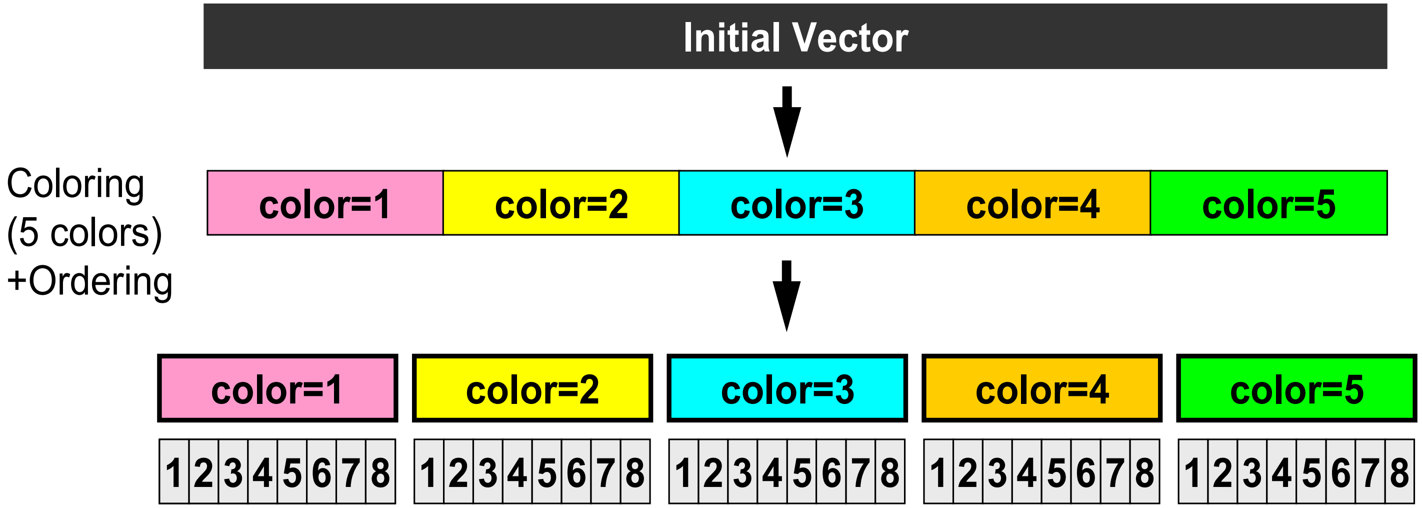
- Material: ICCG solver for sparse matrices derived from FVM applications (Finite Volume Method).
- Parallelization on a single node of Oakleaf-FX (FX10) using OpenMP
  - Data Placement
  - Reordering
- Effects of reordering

Future Directions

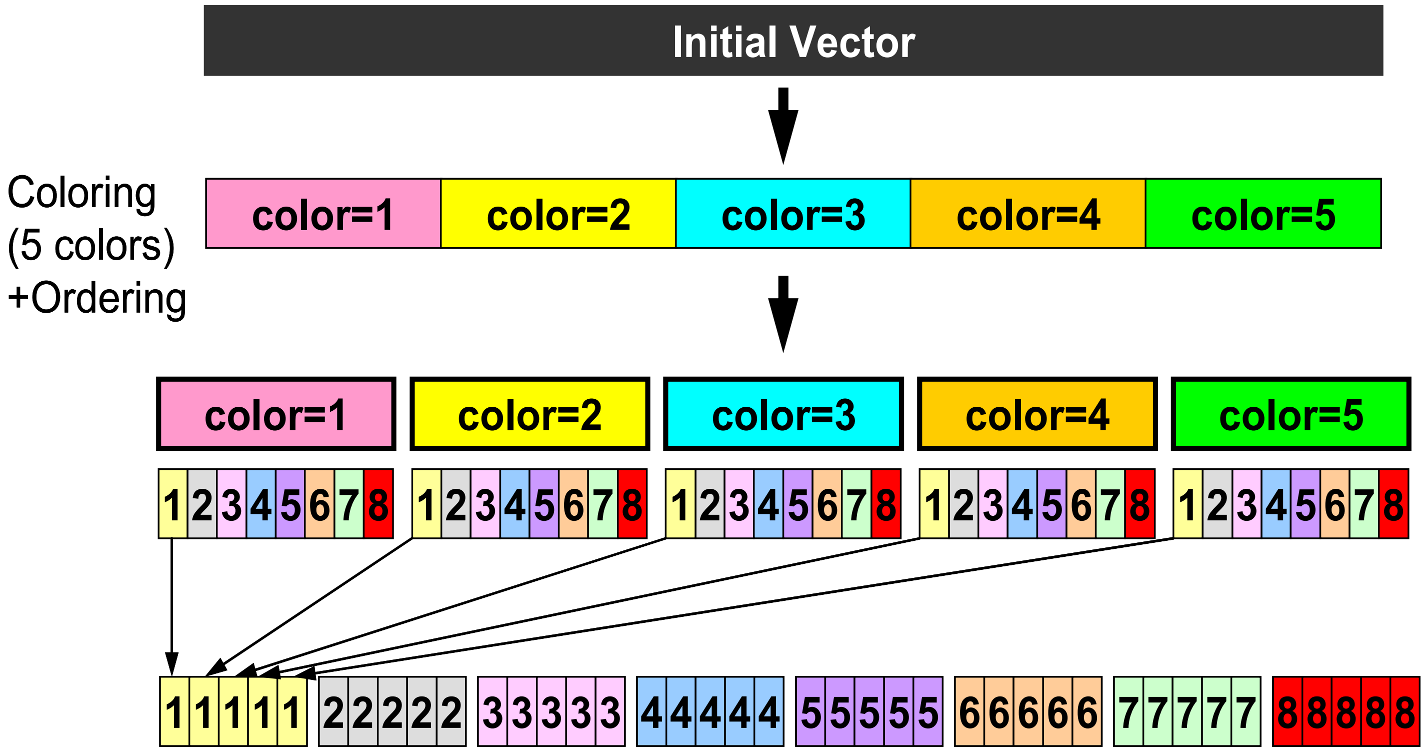
- Gap between performance of CPU & memory
  - BYTE/FLOP
- Multicore/Manycore
  - Intel Xeon/Phi
- Supercomputer system with $>10^5$ cores
  - Exascale：$>10^8$
- Reordering/Ordering
  - Intensity of components of matrices should be also considered (not only the connectivity information)
  - Selection of optimum number of colors: research topic, especially for ill-conditioned problems
- OpenMP/MPI Hybrid -> One of effective choices
  - Optimization for OpenMP is the most critical

Code Name	FX10	MIC	IvyB
Architectures	**Fujitsu SPARC64 IX fx**	**Intel Xeon Phi 5110P (Knights Corner)**	**Intel Xeon E5-2680 v2 (Ivy-Bridge-EP)**
Frequency (GHz)	1.848	1.053	2.80
Core # (Thread #)	16 (16)	60 (240)	10 (20)
Memory Type	DDR3	GDDR5	DDR3
Peak Performance (GFLOPS)	236.5	1,010.9	224.0
RAM (GB)	32	8	64
Peak Memory Bandwidth (GB/s)	85.1	320	59.7
Cache	L1:32KB/core L2:12MB/socket	L1:32KB/core L2:512KB/core	L1:32KB/core L2:256KB/core L3:25MB/socket

Coalesced



Sequential



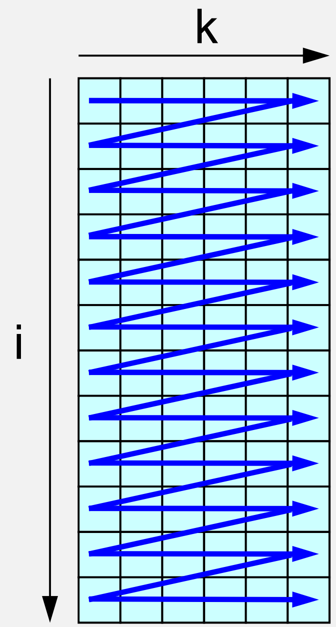
<table>
<tr><th></th><th>Numbering</th><th>Matirx</th><th>Outer Loops</th><th>etc.</th></tr>
<tr><td>AR-0</td><td rowspan="5">Coalesced</td><td>CRS</td><td rowspan="3">Row-wise</td><td></td></tr>
<tr><td>AR-1</td><td rowspan="4">Sliced ELL</td><td></td></tr>
<tr><td>AR-2</td><td>Indirect Access</td></tr>
<tr><td>AC-1</td><td rowspan="2">Column-wise</td><td></td></tr>
<tr><td>AC-2</td><td>Blocking</td></tr>
<tr><td>BR-0</td><td rowspan="4">Sequential</td><td>CRS</td><td rowspan="2">Row-wise</td><td></td></tr>
<tr><td>BR-1</td><td rowspan="3">Sliced ELL</td><td></td></tr>
<tr><td>BC-1</td><td rowspan="2">Column-wise</td><td></td></tr>
<tr><td>BC-2</td><td>Blocking</td></tr>
</table>

ELL: Row-wise

CRS with fixed length

Forward Substitution

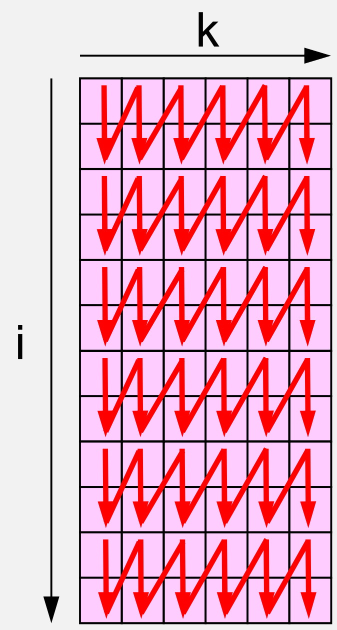
```
!$omp parallel
    do icol= 1, NCOLORtot
!$omp do
    do ip   = 1, PEsmpTOT
      do i= Index(ip-1,icol)+1, Index(ip,icol)
        do k= 1, 6
          Z(i)= Z(i) - AML(k,i)*Z(IAML(k,i))
        enddo
        Z(i)= Z(i) / DD(i)
      enddo
    enddo
    enddo
!omp end parallel
```



ELL: Column-wise

Jagged Diagonal
Forward Substitution

```
!$omp parallel
    do icol= 1, NCOLORtot
!$omp do
    do ip  = 1, PEsmpTOT
      do k= 1, 6
        do i= Index(ip-1,icol)+1, Index(ip,icol)
          Z(i)= Z(i) + AML (i,k)*Z(IAML(i,k))
        enddo
      enddo
      do i= Index(ip-1,icol)+1, Index(ip,icol)
        Z(i)= Z(i) / DD(i)
      enddo
    enddo
    enddo
!omp end parallel
```



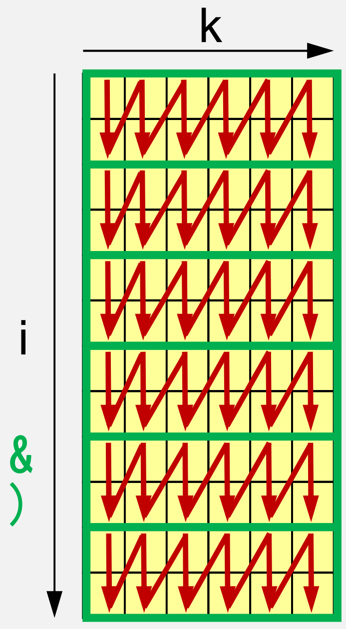
Each Color/Thread

ELL: Column-wise

Jagged Diagonal, Blocked Version

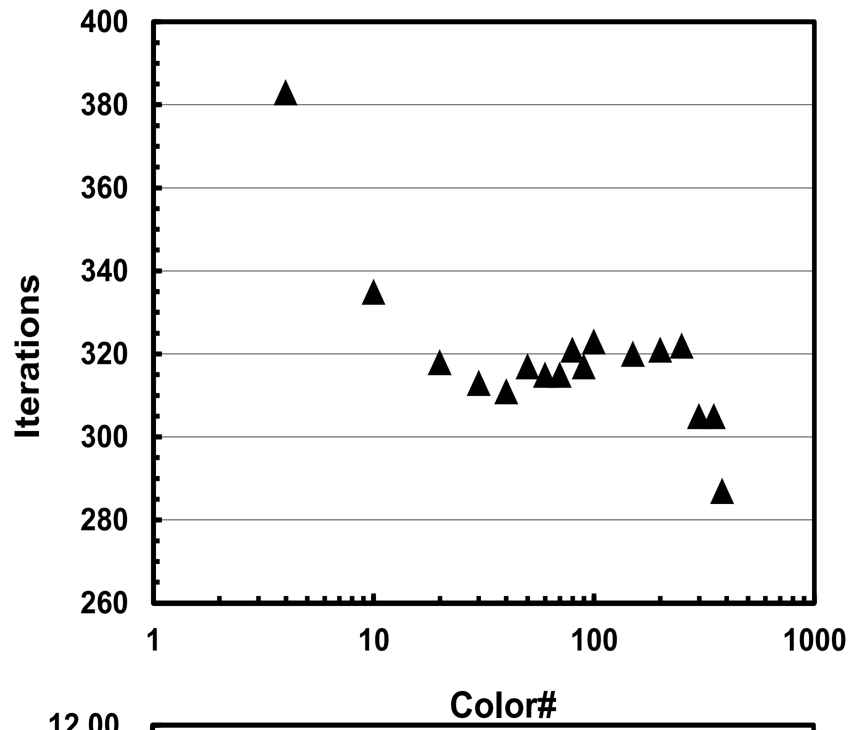
Forward Substitution

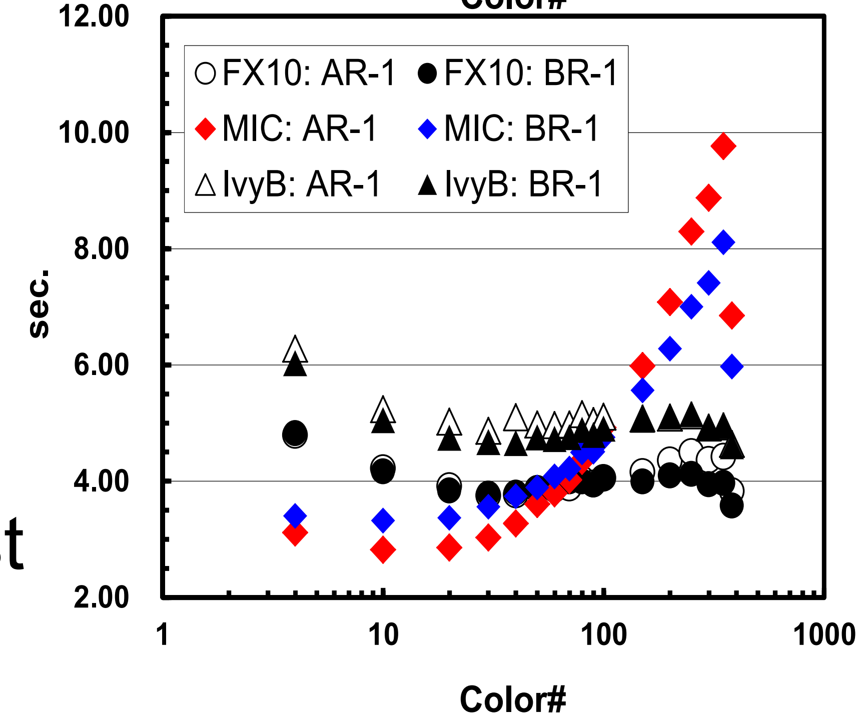
```
!$omp parallel
    do icol= 1, NCOLORtot
!$omp do
    do ip  = 1, PEsmpTOT
      blkID= (ip-1)*NCOLORtot + ip
      do k= 1, 6
      do i= IndexB(ip-1,blkID,icol)+1, &
            IndexB(ip  ,blkID,icol)
        locID= i - IndexB(ip-1,blkID,icol)
        Z(i)= Z(i) +                                    &
              AMLb(locID,k,blkID)* X(IAMLb(locID,k,blkID))
      enddo
      enddo
      do i= IndexB(ip-1,blkID,icol)+1, IndexB(ip  ,blkID,icol)
        Z(i)= Z(i) / DD(i)
      enddo
    enddo
    enddo
!omp end parallel
```



Number of Colors and Comp. Time

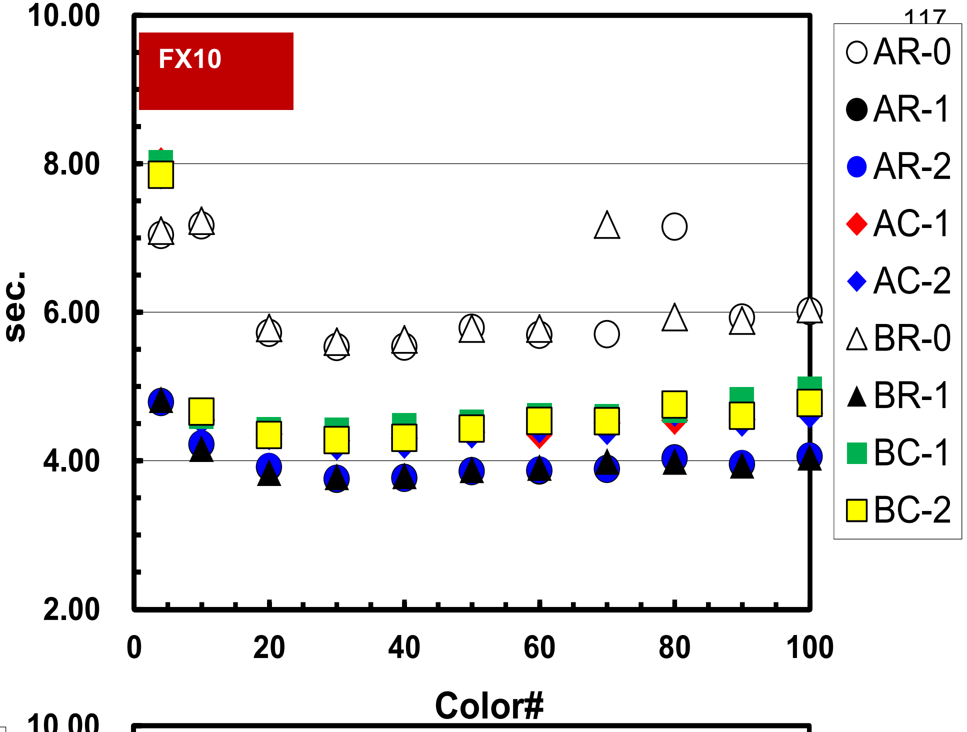
- ELL
  - AR-1: Coalesced + Row-wise
  - BR-1: Sequential + Row-wise
- More colors
  - less iterations
  - more sync. overhead
  - significant in MIC
- Optimum number of color depends on architectures
- FX10, IvyB: RCM is the best

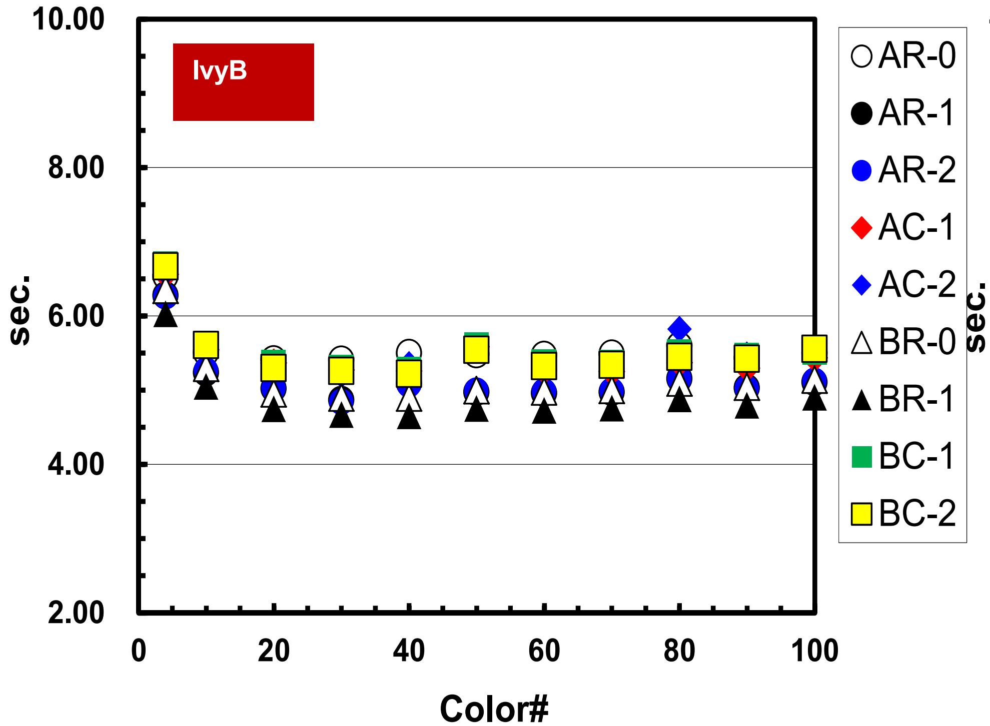


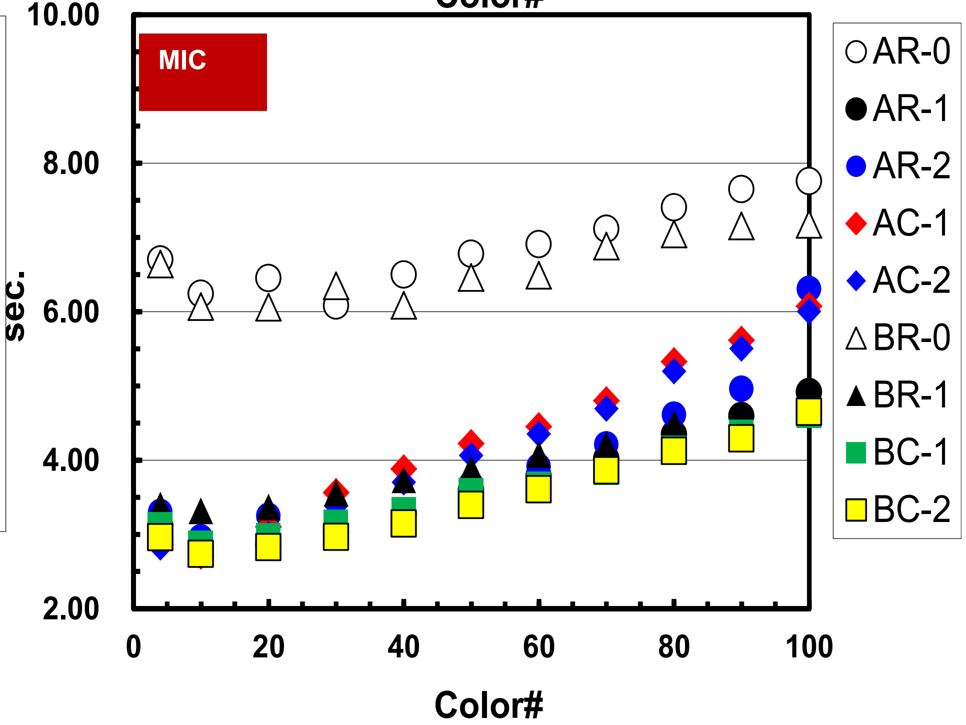


Effect of Number of Colors (~100)

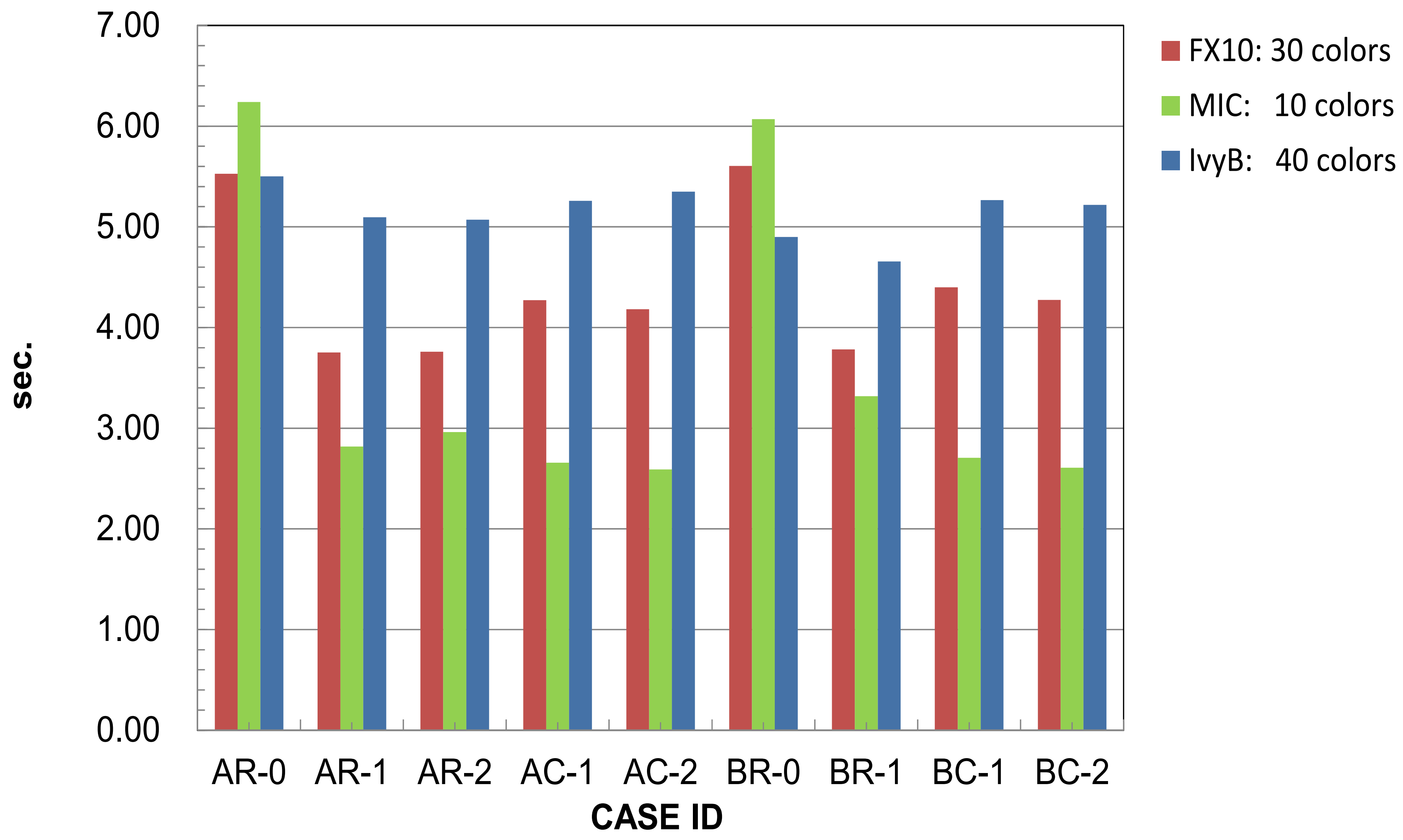
- FX10, MIC: effect of ELL is significant
- IvyB: Effect of CRS/ELL, number of colors: small
  - Out-of-Order ?



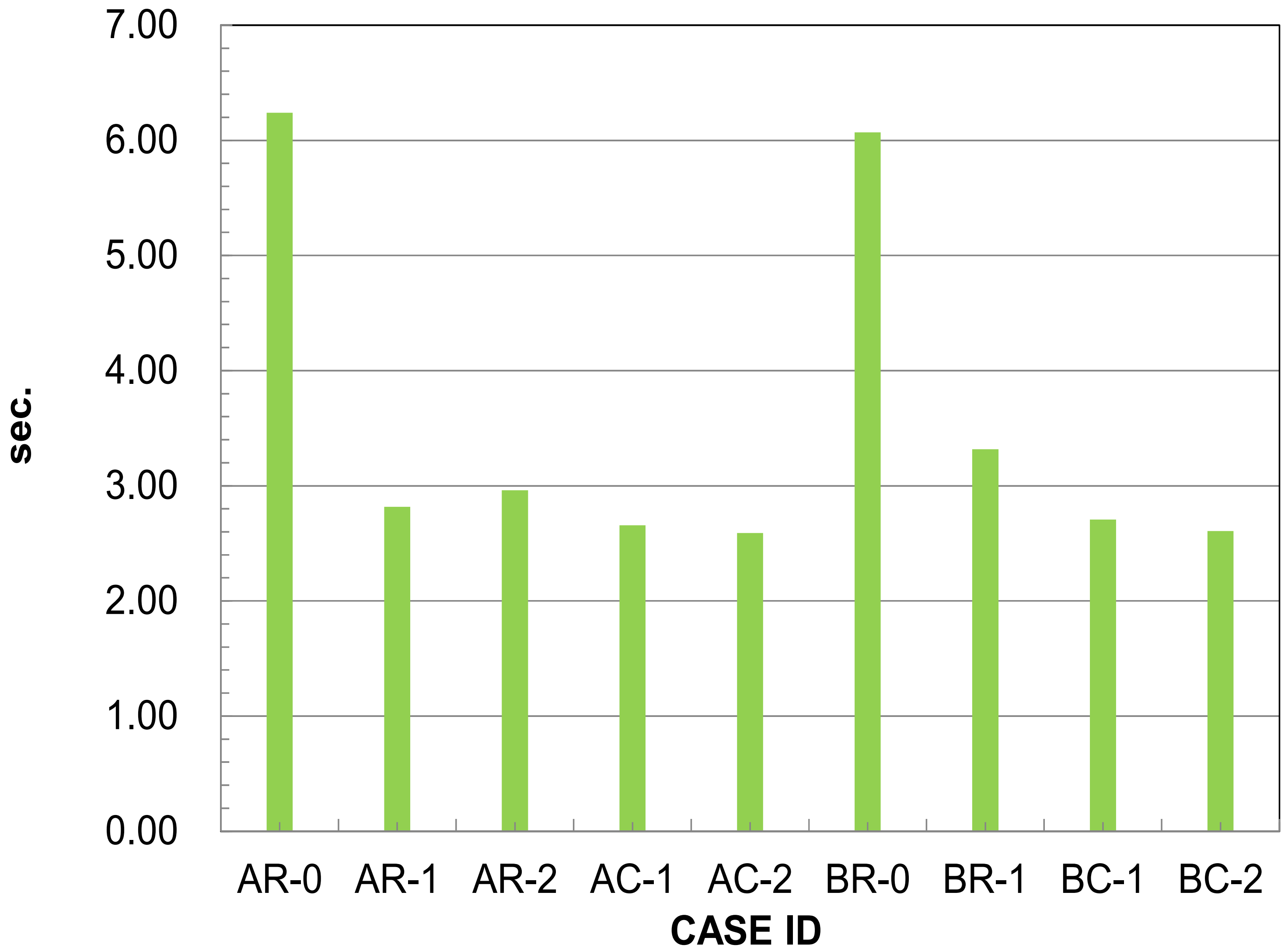




Results (Computation Time for Linear Solver): Down is Good !



Results: MIC



FX10: 30 colors

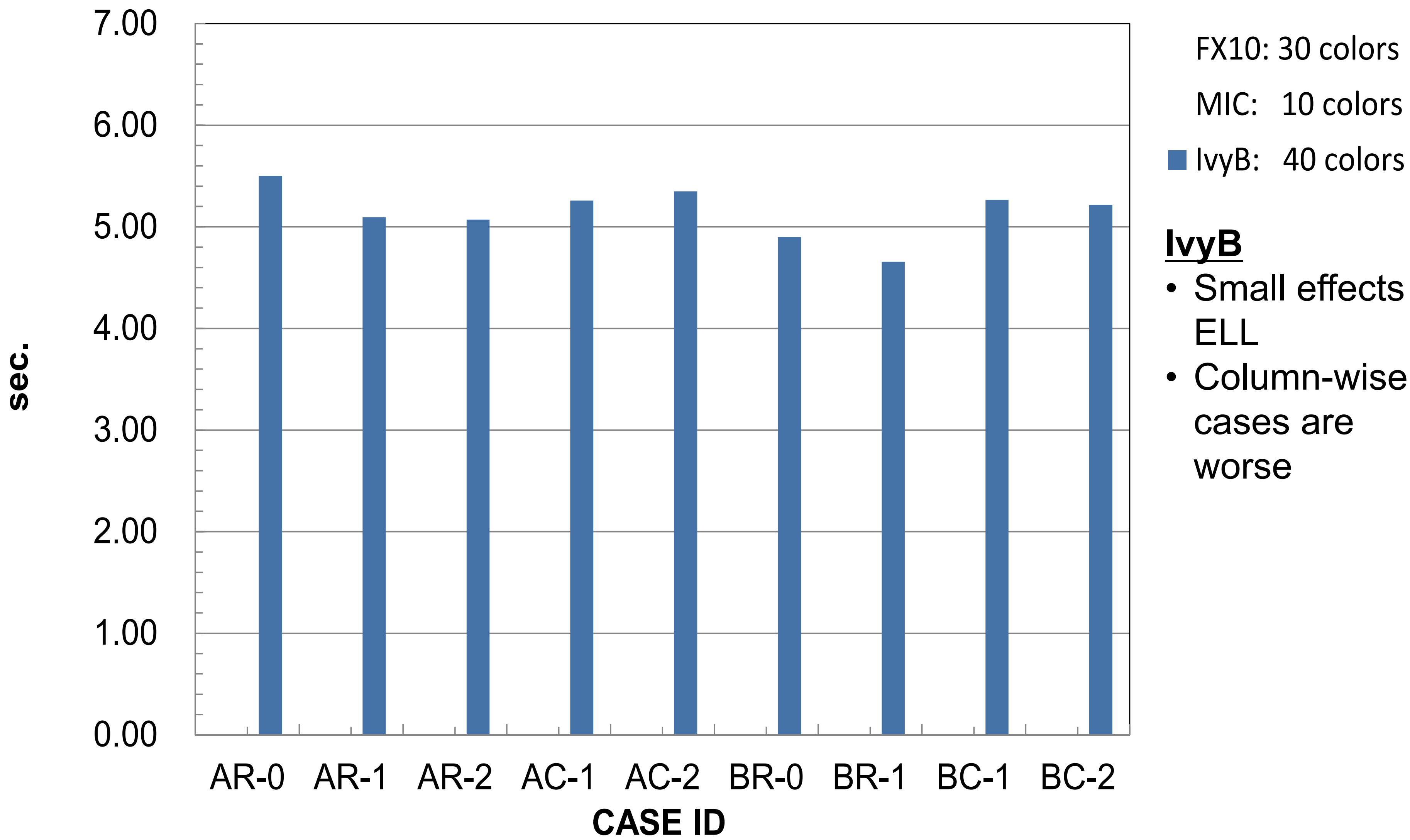
MIC: 10 colors

IvyB: 40 colors

MIC

- Effect of ELL is significant
- Column-wise cases (AC-X, BC-X) are better: Vectorization, SIMD
- Effect of blocking (XC-1=>XC-2)

Results: IvyB



<u>IvyB</u>

- Small effects of ELL
- Column-wise cases are worse