並列有限要素法による
三次元定常熱伝導解析プログラム
OpenMP＋ハイブリッド並列化
C言語編

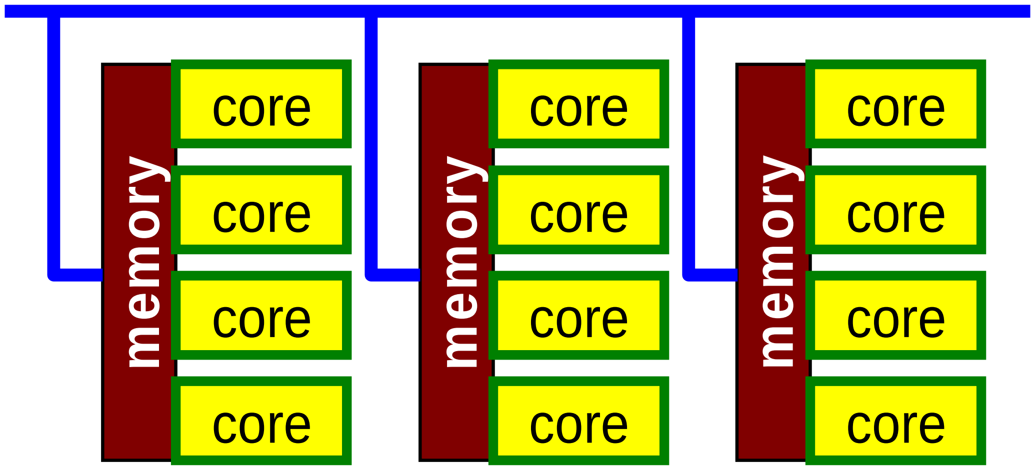
中島 研吾
東京大学情報基盤センター

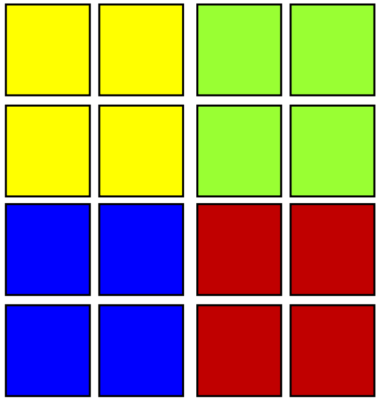
Hybrid並列プログラミング

- スレッド並列＋メッセージパッシング
  - OpenMP+ MPI
  - CUDA + MPI, OpenACC + MPI
- OpenMPがMPIより簡単ということはない
  - データ依存性のない計算であれば，機械的にOpenMP指示文を入れれば良い
  - NUMAになるとより複雑：First Touch Data Placement

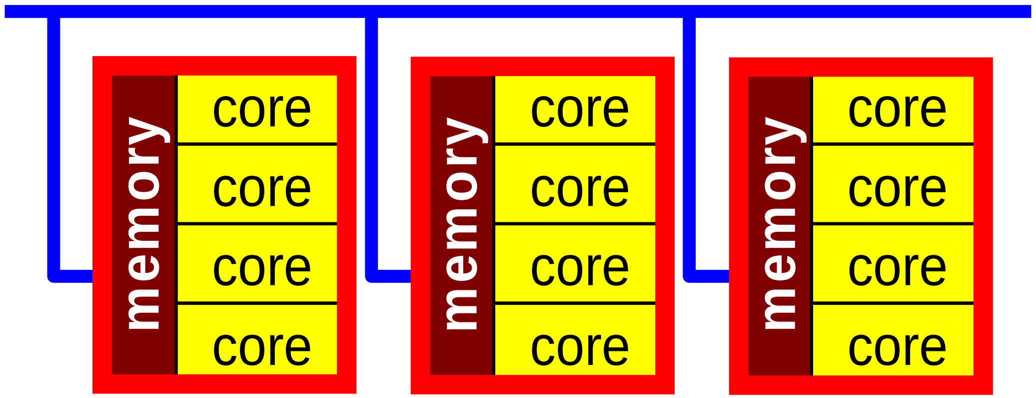
Flat MPI vs. Hybrid

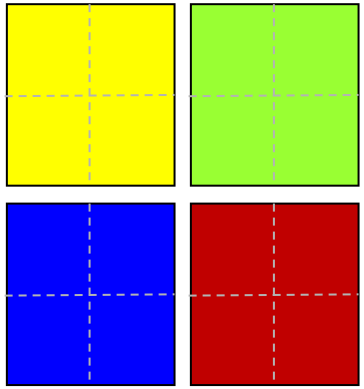
Flat-MPI：Each Core -> Independent

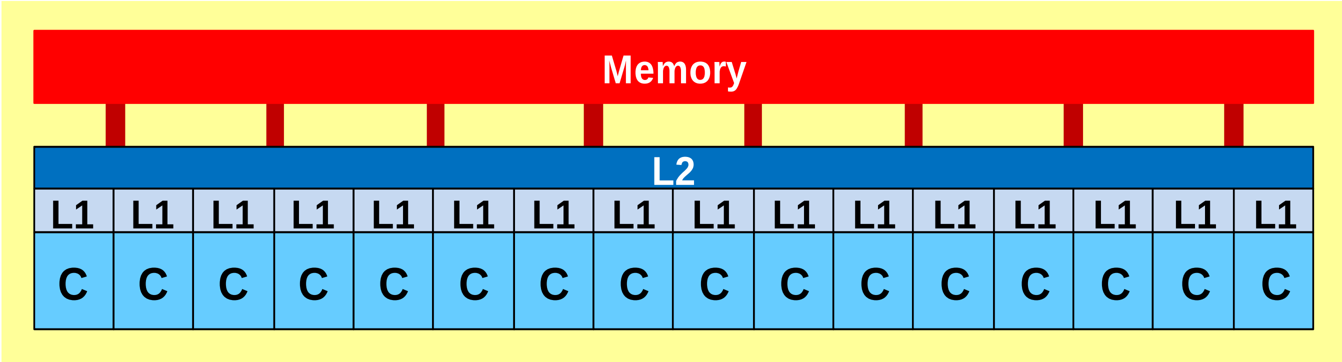




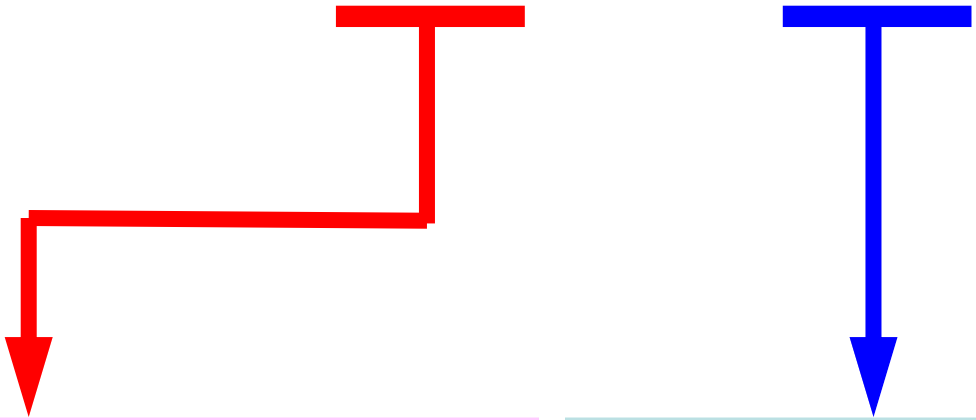
Hybrid：Hierarchal Structure







HB M x N



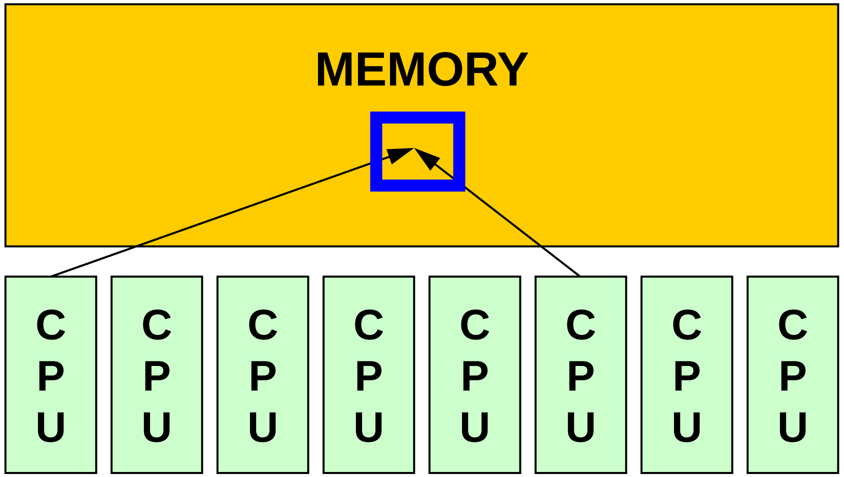
Number of OpenMP threads per a single MPI process

Number of MPI process per a single node

本編の背景

- マイクロプロセッサのマルチコア化，メニーコア化
  - 低消費電力，様々なプログラミングモデル
- OpenMP
  - 指示行（ディレクティヴ）を挿入するだけで手軽に「並列化」ができるため，広く使用されている
  - 様々な解説書
- データ依存性（data dependency）
  - メモリへの書き込みと参照が同時に発生
  - 並列化には，適切なデータの並べ替えを施す必要がある
  - このような対策はOpenMP向けの解説書でも詳しく取り上げられることは余りない：とても面倒くさい
  - この部分は「マルチコアプログラミング」講習会で！
- Hybrid 並列プログラミングモデル

共有メモリ型計算機



- SMP
  - Symmetric Multi Processors
  - 複数のCPU(コア)で同じメモリ空間を共有するアーキテクチュア

OpenMPとは（1/2）

http://www.openmp.org

- 共有メモリ型並列計算機用のDirectiveの統一規格
  - MPIやHPFに比べると遅く1996年頃から活動開始
  - 現在 Ver.4.X
- 背景
  - CrayとSGIの合併（1996）
  - ASCI計画の開始（1995）
    - Accelerated Strategic Computing Initiative (ASCI) -> Advanced Simulation and Computing Program (ASC)
    - ASCI：核実験のシミュレーションによる代替
      - 計算機開発，シミュレーションソフトウェア
    - SMPクラスタにフォーカス
      - ACI Red (Intel)，Blue Pacific/White/Purple/BlueGene (IBM)，Blue Mountain（SGI）
    - SMPクラスタ向けの並列プログラミングの共通API（Application Program Interface）が必要

OpenMPとは（2/2）

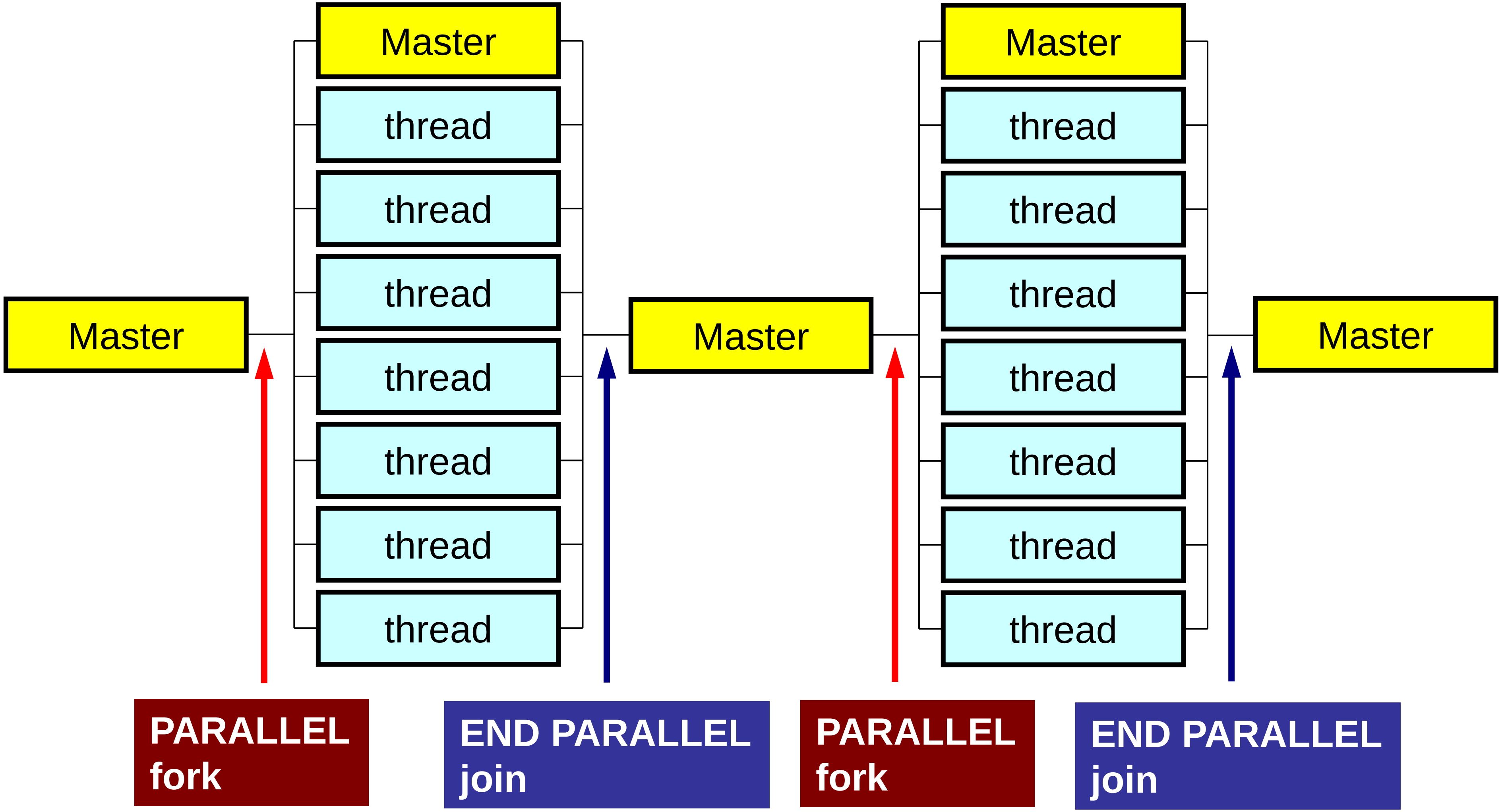
http://www.openmp.org

- 主な計算機ベンダーが集まって OpenMP ARBを結成し，1997年にはもう規格案ができていたそうである
  - SC98ではすでにOpenMPのチュートリアルがあったし，すでにSGI Origin2000でOpenMP-MPIハイブリッドのシミュレーションをやっている例もあった。
- OpenMPはFortan版とC/C++版の規格が全く別々に進められてきた。
  - Ver.2.5で言語間の仕様を統一
- Ver.4.0ではGPU，Intel-MIC等Co-Processor, Accelerator環境での動作も考慮
  - OpenACCに近づいている

OpenMPの概要

- 基本的仕様
  - プログラムを並列に実行するための動作をユーザーが明示
  - OpenMP実行環境は，依存関係，衝突，デッドロック，競合条件，結果としてプログラムが誤った実行につながるような問題に関するチェックは要求されていない。
  - プログラムが正しく実行されるよう構成するのはユーザーの責任である。
- 実行モデル
  - fork-join型並列モデル
    - 当初はマスタスレッドと呼ばれる単一プログラムとして実行を開始し，「PARALLEL」，「END PARALLEL」ディレクティヴの対で並列構造を構成する。並列構造が現れるとマスタスレッドはスレッドのチームを生成し，そのチームのマスタとなる。
  - いわゆる「入れ子構造」も可能であるが，ここでは扱わない

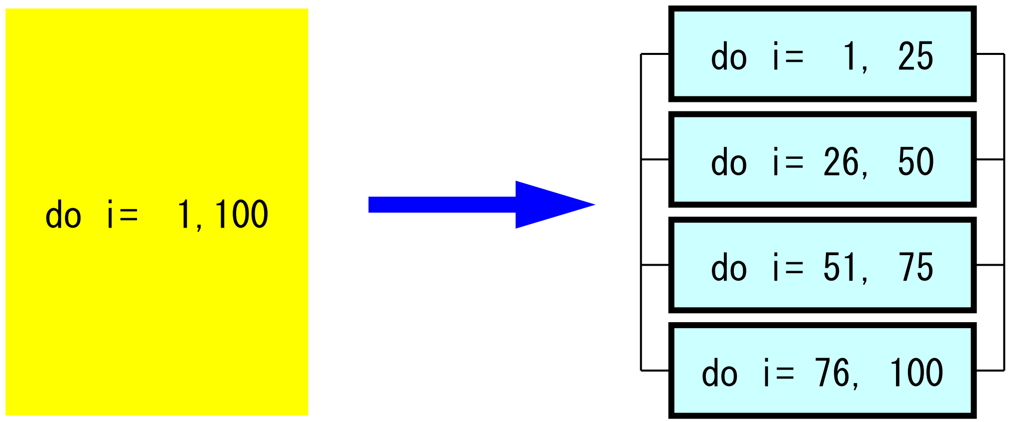
Fork-Join 型並列モデル



スレッド数

- 環境変数 **OMP_NUM_THREADS**
  - 値の変え方
    - bash(.bashrc) **export OMP_NUM_THREADS=8**
    - csh(.cshrc) **setenv OMP_NUM_THREADS 8**

- たとえば，**OMP_NUM_THREADS=4**とすると，以下のように**i=1~100**のループが**4**分割され，同時に実行される。



OpenMPに関連する情報

- OpenMP Architecture Review Board (ARB)
  - http://www.openmp.org
- 参考文献
  - Chandra, R. et al.「Parallel Programming in OpenMP」（Morgan Kaufmann）
  - Quinn, M.J. 「Parallel Programming in C with MPI and OpenMP」（McGrawHill）
  - Mattson, T.G. et al. 「Patterns for Parallel Programming」（Addison Wesley）
  - 牛島「OpenMPによる並列プログラミングと数値計算法」（丸善）
  - Chapman, B. et al. 「Using OpenMP」（MIT Press）
- 富士通他による翻訳：（OpenMP 3.0）必携！
  - http://www.openmp.org/mp-documents/OpenMP30spec-ja.pdf

OpenMPの特徴

- ディレクティヴ（指示行）の形で利用
  - 挿入直後のループが並列化される
  - コンパイラがサポートしていなければ，コメントとみなされる

OpenMP/Directives Array Operations

Simple Substitution

```
#pragma omp parallel for private (i)
 for (i=0; i<N; i++) {
      X[i] = 0.0;
      W[0][i] = 0.0;
      W[1][i] = 0.0;
      W[2][i] = 0.0;
 }
```

Dot Products

```
RHO = 0.0;
#pragma omp parallel for private (i) 
 reduction (+:RHO)
 for(i=0; i<N; i++) {
      RHO += W[R][i] * W[Z][i];
 }
```

DAXPY

```
#pragma omp parallel for private (i)
 for (i=0; i<N; i++) {
     Y[i] = Y[i} + alpha*X[i];
 }
```

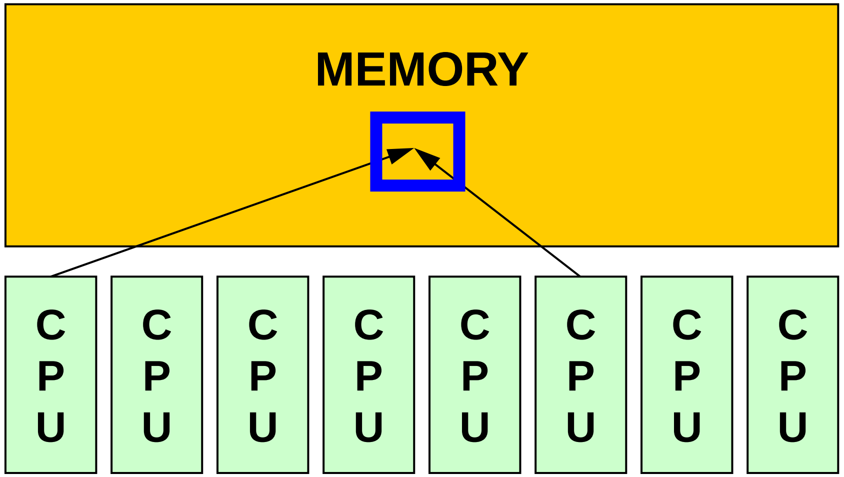
OpenMP/Direceives Matrix/Vector Products

```
#pragma omp parallel for private (i,VAL,j)
 for(i=0; i<N; i++) {
     VAL = D[i] * W[P][i];
     for(j=indexL[i]; j<indexL[i+1]; j++) {
          VAL += AL[j] * W[P][itemL[j]-1];
     }

     for(j=indexU[i]; j<indexU[i+1]; j++) {
          VAL += AU[j] * W[P][itemU[j]-1];
     }
     W[Q][i] = VAL;
 }
```

OpenMPの特徴

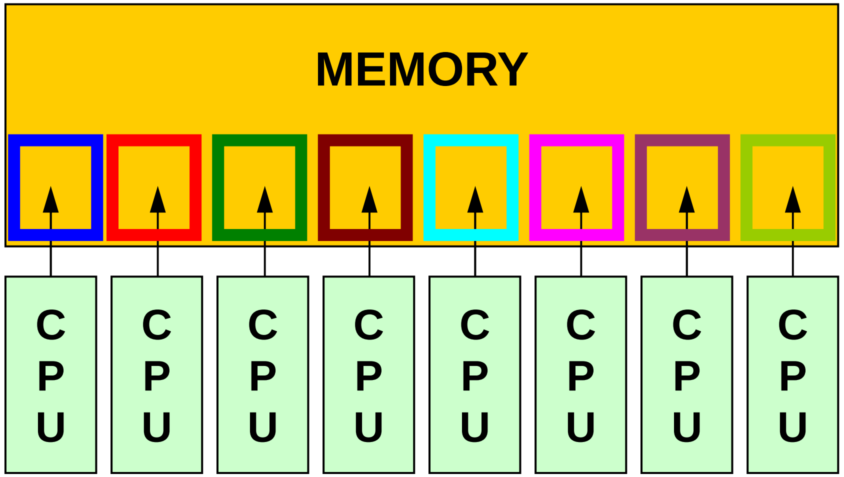
- ディレクティヴ（指示行）の形で利用
  - 挿入直後のループが並列化される
  - コンパイラがサポートしていなければ，コメントとみなされる
- 何も指定しなければ，何もしない
  - 「自動並列化」，「自動ベクトル化」とは異なる。
  - 下手なことをするとおかしな結果になる：ベクトル化と同じ
  - データ分散等（Ordering）は利用者の責任
- 共有メモリユニット内のプロセッサ数に応じて，「Thread」が立ち上がる
  - 「Thread」：MPIでいう「プロセス」に相当する。
  - 普通は「Thread数＝共有メモリユニット内プロセッサ数，コア数」であるが最近のアーキテクチャではHyper Threading（HT）がサポートされているものが多い（1コアで2-4スレッド）

メモリ競合（Memory Contention）



- 複雑な処理をしている場合，複数のスレッドがメモリ上の同じアドレスにあるデータを同時に更新する可能性がある。
  - 複数のCPUが配列の同じ成分を更新しようとする。
  - メモリを複数のコアで共有しているためこのようなことが起こりうる。
  - 場合によっては答えが変わる

メモリ競合（Memory Contention）（続き）



- 本演習で扱っている例は，そのようなことが生じないよう，各スレッドが同時に同じ成分を更新するようなことはないようにする。
  - これはユーザーの責任でやること，である。
- データ依存性（Data Dependency）
- 多くのコア数（スレッド数）が増えるほど，メモリへの負担が増えて，処理速度は低下する（メモリ飽和）。

OpenMPの特徴（続き）

- 基本は「#pragma omp parallel for」
- 変数について，グローバル/sharedな変数と，Thread内でローカルな「private」な変数に分けられる。
  - デフォルトは「global/shared」
  - 内積を求める場合は「reduction」を使う

```
     VAL= 0.0;
#pragma omp parallel for private (i,ip)
 reduction(+:VAL)
    for(ip=0; ip<PEsmpTOT; ip++){
        for (i=INDEX[ip]; i<INDEX[ip+1]; i++){
           VAL= VAL + W[R][i] * W[Z][i];
        }
    }
```

W(:,:), R, Z, PEsmpTOT などはグローバル変数

FORTRANとC

```
use omp_lib
...
!$omp parallel do shared(n,x,y) private(i)
      do i= 1, n
        x(i)= x(i) + y(i)
      enddo
!$ omp end parallel do
```

```
#include <omp.h>
...
{
  #pragma omp parallel for default(none) shared(n,x,y) private(i)

    for (i=0; i<n; i++)
        x[i] += y[i];
}
```

本講義における方針

- OpenMPは多様な機能を持っているが，それらの全てを逐一教えることはしない。
  - 講演者も全てを把握，理解しているわけではない。
- 数値解析に必要な最低限の機能のみ学習する。
  - 具体的には，講義で扱っているICCG法によるポアソン方程式ソルバーを動かすために必要な機能のみについて学習する
  - それ以外の機能については，自習，質問のこと（全てに答えられるとは限らない）。
- MPIと同じ

最初にやること

- use omp_lib　FORTRAN
- #include <omp.h>　C

- 様々な環境変数，インタフェースの定義（OpenMP3.0以降でサポート）

OpenMPディレクティヴ(FORTRAN)

```
sentinel directive_name [clause[[,] clause]…]
```

- 大文字小文字は区別されない。
- sentinel
  - 接頭辞
  - FORTRANでは「!$OMP」, 「C$OMP」, 「*$OMP」, 但し自由ソース形式では「!$OMP」のみ。
  - 継続行にはFORTRANと同じルールが適用される。以下はいずれも「!$OMP PARALLEL DO SHARED(A,B,C)」

```
!$OMP PARALLEL DO
!$OMP+SHARED (A,B,C)
```

```
!$OMP PARALLEL DO &
!$OMP SHARED (A,B,C)
```

OpenMPディレクィヴ（C）

```
#pragma omp directive_name [clause[[,] clause]…]
```

- 継続行は「＼」
- 小文字を使用（変数名以外）

```
#pragma omp parallel for shared (a,b,c)
```

PARALLEL DO/for

```
!$OMP PARALLEL DO[clause[[,] clause] … ]
   (do_loop)
!$OMP END PARALLEL DO
```

```
#pragma omp parallel for [clause[[,] clause] … ]
   (for_loop)
```

- 多重スレッドによって実行される領域を定義し，DOループの並列化を実施する。
- 並び項目（clause）：よく利用するもの
  - PRIVATE（list）
  - SHARED（list）
  - DEFAULT（PRIVATE|SHARED|NONE）
  - REDUCTION（{operation|intrinsic}: list）

REDUCTION

```
REDUCTION ({operator|instinsic}: list)
```

```
reduction ({operator|instinsic}: list)
```

- 「MPI_REDUCE」のようなものと思えばよい
- Operator
  - ＋, *, -, .AND., .OR., .EQV., .NEQV.
- Intrinsic
  - MAX, MIN, IAND, IOR, IEQR

実例A1：簡単なループ

```
#pragma omp parllel for
 for(i=0; i<N; i++){
      B[i]= (A[i] + B[i]) * 0.50;
 }
```

- ループの繰り返し変数（ここでは「i」）はデフォルトでprivateなので，明示的に宣言は不要。
- 「END PARALLEL DO」は省略可能。
  - C言語ではそもそも存在しない

実例A2：REDUCTION

```
#pragma omp parallel default(private) reduction(+:A,B)
 for(i=0; i<N; i++){
     err= work(Alocal, Blocal);
     A= A + Alocal;
     B= B + Blocal;
 }
```

OpenMP使用時に呼び出すことのできる関数群

関数名	内容
int omp_get_num_threads (void)	スレッド総数
int omp_get_thread_num (void)	自スレッドのID
double omp_get_wtime (void)	MPI_Wtimeと同じ
void omp_set_num_threads (int num_threads) call omp_set_num_threads (num_threads)	スレッド数設定

OpenMP for Dot Products

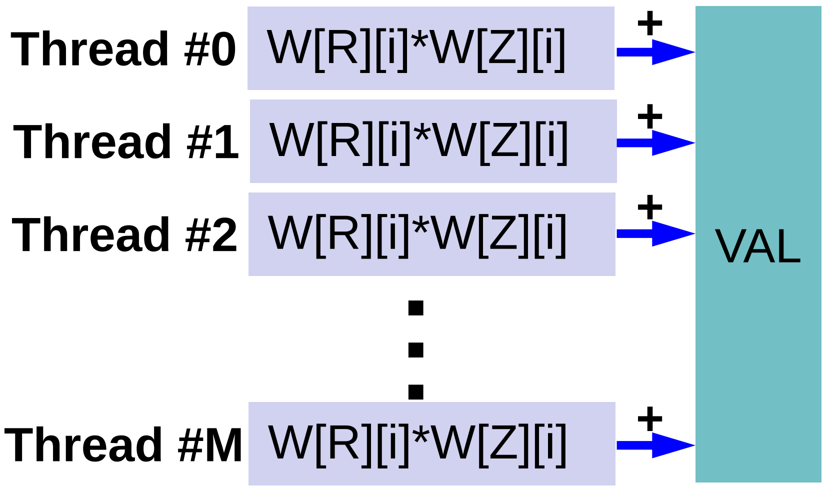
```
VAL= 0.0;
for(i=0; i<N; i++){
  VAL= VAL + W[R][i] * W[Z][i];
}
```

OpenMP for Dot Products

```
    VAL= 0.0;
    for(i=0; i<N; i++){
      VAL= VAL + W[R][i] * W[Z][i];
    }
```

```
    VAL= 0.0;
#pragma omp parallel for private (i) reduction(+:VAL)
    for(i=0; i<N; i++){
      VAL= VAL + W[R][i] * W[Z][i];
    }
```

Directives are just inserted.



OpenMP for Dot Products

```
    VAL= 0.0;
    for(i=0; i<N; i++){
      VAL= VAL + W[R][i] * W[Z][i];
    }
```

```
    VAL= 0.0;
#pragma omp parallel for private (i) reduction(+:VAL)
    for(i=0; i<N; i++){
      VAL= VAL + W[R][i] * W[Z][i];
    }
```

OpenMPディレクティヴの挿入
これでも並列計算は可能

```
     VAL= 0.0;
#pragma omp parallel for private (i,ip)
 reduction(+:VAL)
    for(ip=0; ip<PEsmpTOT; ip++){
        for (i=INDEX[ip]; i<INDEX[ip+1]; i++){
           VAL= VAL + W[R][i] * W[Z][i];
        }
    }
```

多重ループの導入
PEsmpTOT:スレッド数
あらかじめ「INDEX(:)」を用意しておく
より確実に並列計算実施
(別に効率がよくなるわけでは無い)

OpenMP for Dot Products

```
    VAL= 0.0;
    for(i=0; i<N; i++){
      VAL= VAL + W[R][i] * W[Z][i];
    }
```

```
    VAL= 0.0;
#pragma omp parallel for private (i) reduction(+:VAL)
    for(i=0; i<N; i++){
      VAL= VAL + W[R][i] * W[Z][i];
    }
```

OpenMPディレクティヴの挿入
これでも並列計算は可能

```
     VAL= 0.0;
#pragma omp parallel for private (i,ip)
 reduction(+:VAL)
    for(ip=0; ip<PEsmpTOT; ip++){
        for (i=INDEX[ip]; i<INDEX[ip+1]; i++){
           VAL= VAL + W[R][i] * W[Z][i];
        }
    }
```

多重ループの導入
PEsmpTOT:スレッド数
あらかじめ「INDEX(:)」を用意しておく
より確実に並列計算実施

PEsmpTOT個のスレッドが立ち上がり，並列に実行

OpenMP for Dot Products

```
 VAL= 0.0;
#pragma omp parallel for private (i,ip)
 reduction(+:VAL)
    for(ip=0; ip<PEsmpTOT; ip++){
        for (i=INDEX[ip]; i<INDEX[ip+1]; i++){
           VAL= VAL + W[R][i] * W[Z][i];
        }
    }
```

多重ループの導入
PEsmpTOT：スレッド数
あらかじめ「INDEX[:]」を用意しておく
より確実に並列計算実施

PEsmpTOT個のスレッドが立ち上がり，並列に実行

各要素が計算されるスレッドを指定できる

e.g.: N=100, PEsmpTOT=4

INDEX[0]= 0
INDEX[1]= 25
INDEX[2]= 50
INDEX[3]= 75
INDEX[4]= 100

Matrix-Vector Multiply

```
for(i=0; i<N; i++) {
    VAL = D[i] * W[P][i];
    for(j=indexL[i]; j<indexL[i+1]; j++) {
        VAL += AL[j] * W[P][itemL[j]-1];
    }

    for(j=indexU[i]; j<indexU[i+1]; j++) {
        VAL += AU[j] * W[P][itemU[j]-1];
    }
    W[Q][i] = VAL;
}
```

Matrix-Vector Multiply

```
#pragma omp parallel for private(ip,i,VAL,j)
 for (ip=0; ip<PEsmpTOT; ip++){
      for(i=SMPindexG[ip]; i<SMPindexG[ip+1]; i++){
            VAL = D[i] * W[P][i];
            for(j=indexL[i]; j<indexL[i+1]; j++) {
                  VAL += AL[j] * W[P][itemL[j]-1];
            }

            for(j=indexU[i]; j<indexU[i+1]; j++) {
                  VAL += AU[j] * W[P][itemU[j]-1];
            }
            W[Q][i] = VAL;
      }
 }
```

Matrix-Vector Multiply: Other Approach

This is rather better for GPU and (very) many-core architectures: simpler structure of loops

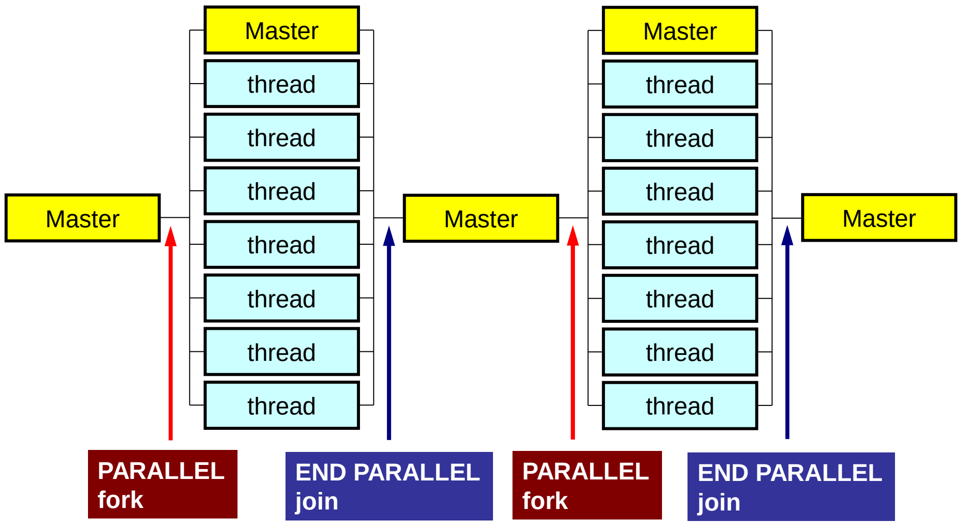
```
#pragma omp parallel for private(i,VAL,j)
 for(i=0; i<N; i++){
          VAL = D[i] * W[P][i];
          for(j=indexL[i]; j<indexL[i+1]; j++) {
              VAL += AL[j] * W[P][itemL[j]-1];
          }

          for(j=indexU[i]; j<indexU[i+1]; j++) {
              VAL += AU[j] * W[P][itemU[j]-1];
          }
          W[Q][i] = VAL;
 }
```

omp parallel (do)

- omp parallel-omp end parallelはそのたびにスレッドを生成，消滅させる：fork-join
- ループが連続するとオーバーヘッドになる。
- omp parallel + omp do/omp for

```
#pragma omp parallel ...

#pragma omp for {

...
#pragma omp for {
```

```
!$omp parallel ...

!$omp do
    do i= 1, N
...
!$omp do
    do i= 1, N
...
!$omp end parallel  required
```



課題

- CGソルバー（solver_CG，solver_SR）のOpenMPによるマルチスレッド化，Hybrid並列化
- 行列生成部（mat_ass_main，mat_ass_bc）のマルチスレッド化，Hybrid並列化

- 問題サイズを変更して計算を実施してみよ
- Hybridでノード内スレッド数を変化させてみよ
  - OMP_NUM_THREADS

OpenMP（Only Solver）（F･C）

```
>$ cd /lustre/gt00/t00XYZ/pfem3d/src1
>$ make
>$ cd ../run
>$ ls sol1
       sol1

>$ cd ../pmesh

<Parallel Mesh Generation>

>$ cd ../run

<modify go1.sh>

>$ qsub go1.sh
```

Makefile（C）

```
CC      = mpiicc
LIB_DIR=
INC_DIR=
OPTFLAGS= -O3 -axCORE-AVX512 -align -qopenmp
LIBS =
LFLAGS=
#
TARGET = ../run/sol1
default: $(TARGET)
OBJS =¥
        test1.o¥...

$(TARGET): $(OBJS)
        $(CC) $(OPTFLAGS) -o $@ $(OBJS) $(LFLAGS)
.c.o:
        $(CC) $(OPTFLAGS) -c  $*.c
clean:
        /bin/rm -f *.o $(TARGET) *~ *.mod
```

How to apply multi-threading

- CG Solver
  - OpenMP指示文を挿入するのみ
  - ILU/IC 前処理の場合はもっと難しい
- MAT_ASS (mat_ass_main, mat_ass_bc)
  - データ依存性あり
  - 複数要素による1節点への足し込みが並列計算時に同時に発生することを避ける必要がある
    - 答えが変わる，もしくはDead Lockが生じる可能性がある
  - 色づけ：Coloring
    - 同じ色に彩色された要素は節点を共有しない
    - 同じ色の要素には並列計算が可能
    - 本問題の場合，三次元では8色，二次元では4色必要
    - 色づけ部分の計算はexpensive：並列化困難

C(solver_CG)

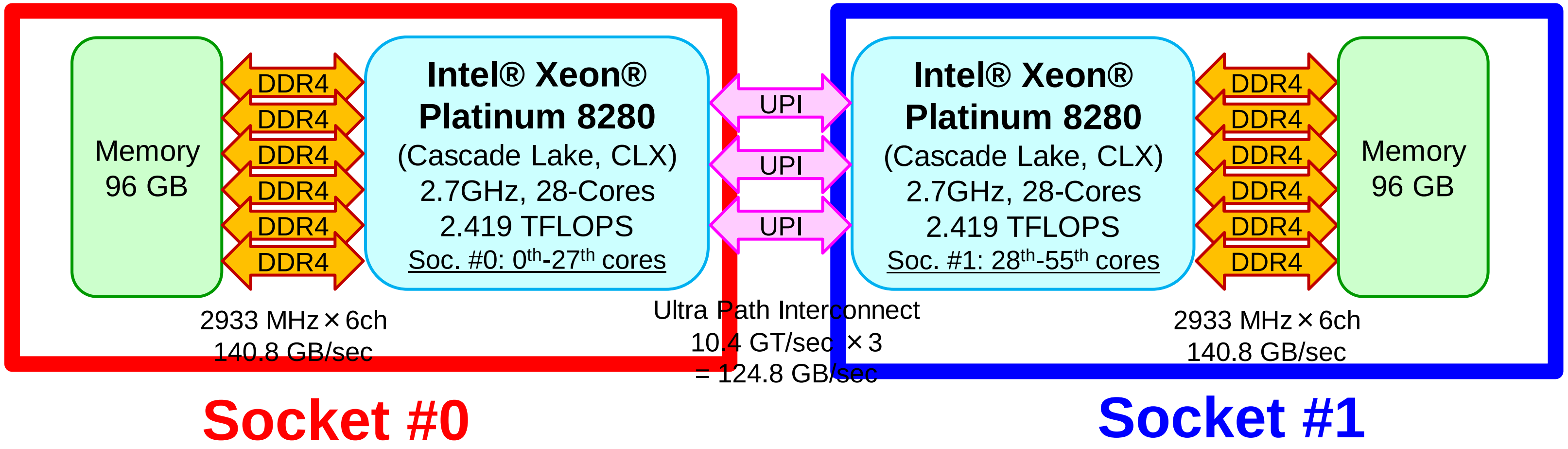
```
#pragma omp parallel for private (i)
    for(i=0;i<N;i++){
      X [i]   +=  ALPHA *WW[P][i];
      WW[R][i]+= -ALPHA *WW[Q][i];
    }
```

```
    DNRM20= 0.e0;
#pragma omp parallel for private (i) reduction (+:DNRM20)
    for(i=0;i<N;i++){
      DNRM20+=WW[R][i]*WW[R][i];
    }
```

```
#pragma omp parallel for private (j,i,k,WVAL)
    for( j=0;j<N;j++){
      WVAL= D[j] * WW[P][j];
      for(k=indexLU[j];k<indexLU[j+1];k++){
        i=itemLU[k];
        WVAL+= AMAT[k] * WW[P][i];
    }
    WW[Q][j]=WVAL;
```

solver_SR (send)

```
 for( neib=1;neib<=NEIBPETOT;neib++){
    istart=EXPORT_INDEX[neib-1];
    inum  =EXPORT_INDEX[neib]-istart;
#pragma omp parallel for private (k,ii)
    for( k=istart;k<istart+inum;k++){
      ii= EXPORT_ITEM[k];
      WS[k]= X[ii-1];
    }
    MPI_Isend(&WS[istart],inum,MPI_DOUBLE,
              NEIBPE[neib-1],0,MPI_COMM_WORLD,&req1[neib-1]);
  }
```



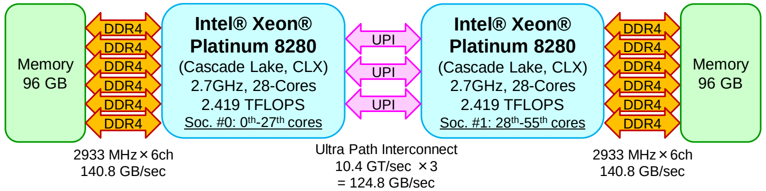
HB M x N

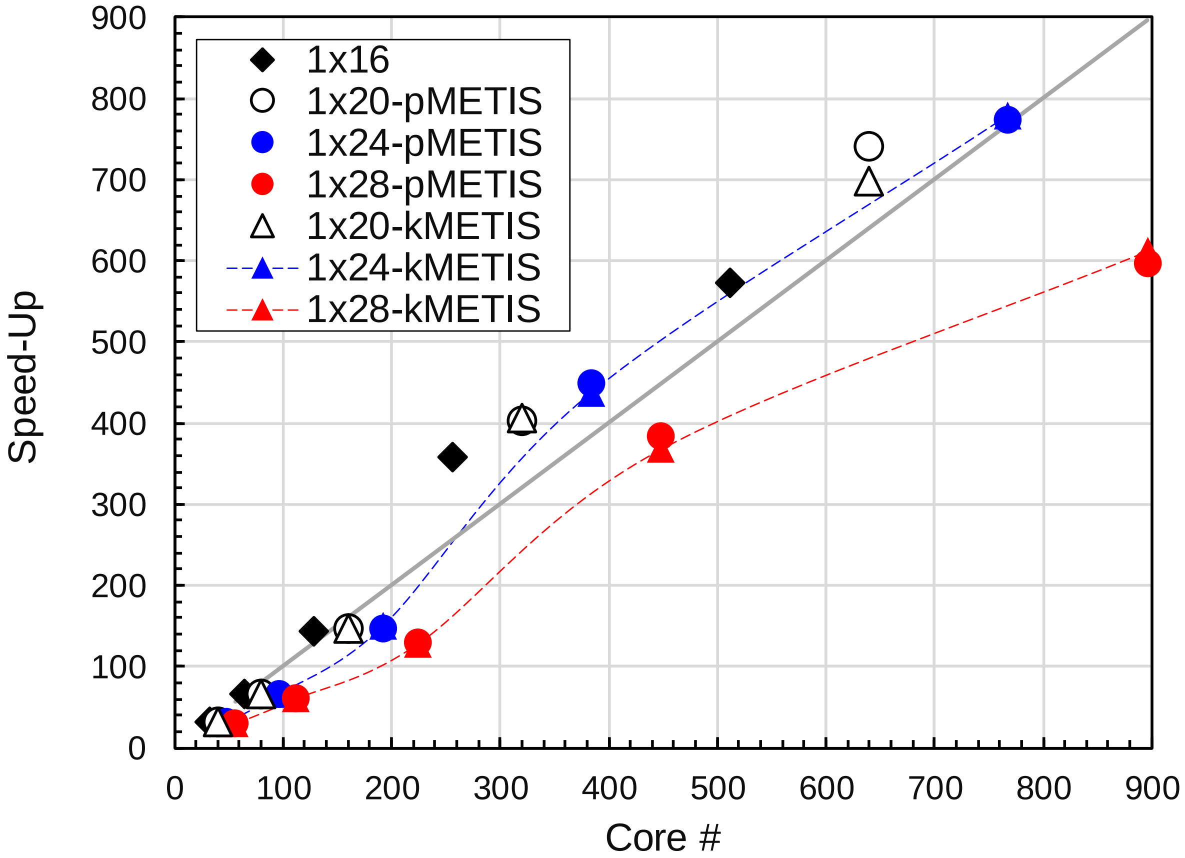
Number of OpenMP threads per a single MPI process

Number of MPI process per a single “socket”

Example: Strong Scaling: Fortran Flat MPI

- 128×128×128 nodes, 2,097,152 DOF
- 32～896 cores, Linear Solver





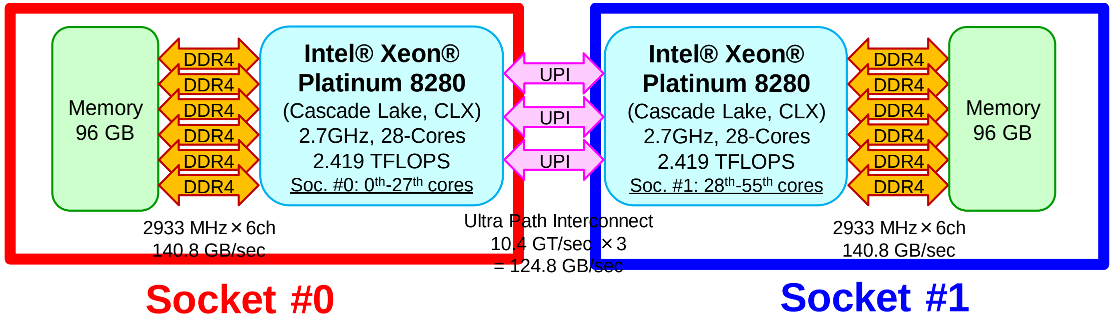
- ◆ 1x16 proc's for each Socket 1x16x2 (=32) for each Node
- ○ 1x20x2 (=40), pMETIS
- ● 1x24x2 (=48), pMETIS
- ● 1x28x2 (=56), pMETIS
- △ 1x20x2 (=40), kMETIS
- ▲ 1x24x2 (=48), kMETIS
- ▲ 1x28x2 (=56), kMETIS

Performance of ◆ without NUMA at 32-cores= 32.0

HB (2or3) x 8, 8-nodes, 128-proc's

t02x08x2.sh t03x08x2.sh

```
mesh.inp
128 128 128
  8   4   4
pcube
```



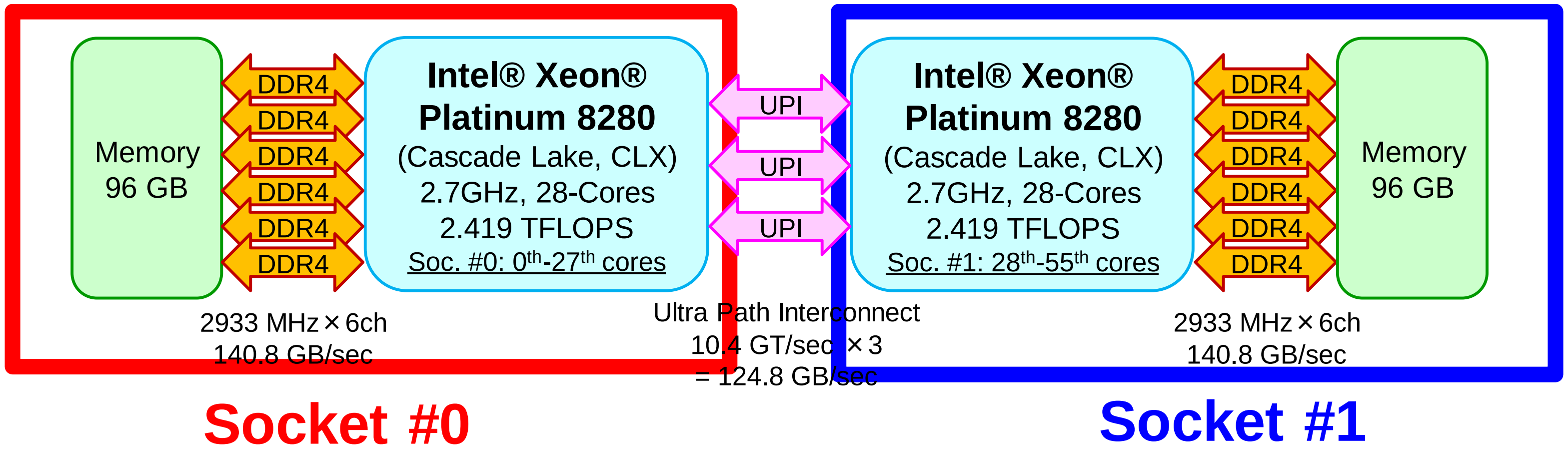
t02x08x2.sh, HB 2x8
16/28 cores on ea. Soc.
2-threads x 8-proc's

```
#!/bin/sh
#PJM -N "HB02x08x2"
#PJM -L rscgrp=lecture9
#PJM -L node=8
#PJM --mpi proc=128
#PJM --omp thread=2
#PJM -L elapse=00:15:00
#PJM -g gt39
#PJM -j
#PJM -e err
#PJM -o t02x08x2_0001.lst

mpiexec.hydra -n ${PJM_MPI_PROC} ./sol1
mpiexec.hydra -n ${PJM_MPI_PROC} numactl -l ./sol1

export KMP_AFFINITY=granularity=fine,compact
mpiexec.hydra -n ${PJM_MPI_PROC} ./sol1
mpiexec.hydra -n ${PJM_MPI_PROC} numactl -l ./sol1
```

t03x08x2.sh, HB 3x8
24/28 cores on ea. Soc.
3-threads x 8-proc's

```
#!/bin/sh
#PJM -N "HB03x08x2"
#PJM -L rscgrp=lecture9
#PJM -L node=8
#PJM --mpi proc=128
#PJM --omp thread=3
#PJM -L elapse=00:15:00
#PJM -g gt39
#PJM -j
#PJM -e err
#PJM -o t03x08x2_0001.lst

mpiexec.hydra -n ${PJM_MPI_PROC} ./sol1
mpiexec.hydra -n ${PJM_MPI_PROC} numactl -l ./sol1

export KMP_AFFINITY=granularity=fine,compact
mpiexec.hydra -n ${PJM_MPI_PROC} ./sol1
mpiexec.hydra -n ${PJM_MPI_PROC} numactl -l ./sol1
```

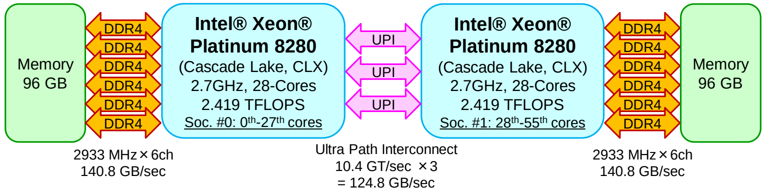
- export KMP_AFFINITY=granularity=fine,compact
  - 各ノードで使用するコア数が56より非常に少ない場合は，Socket#0のコアが優先的に使われる
  - **HB M×NのM（各MPIプロセスにおけるOpenMPスレッド数）が比較的大きい場合には性能を改善する可能性がある**

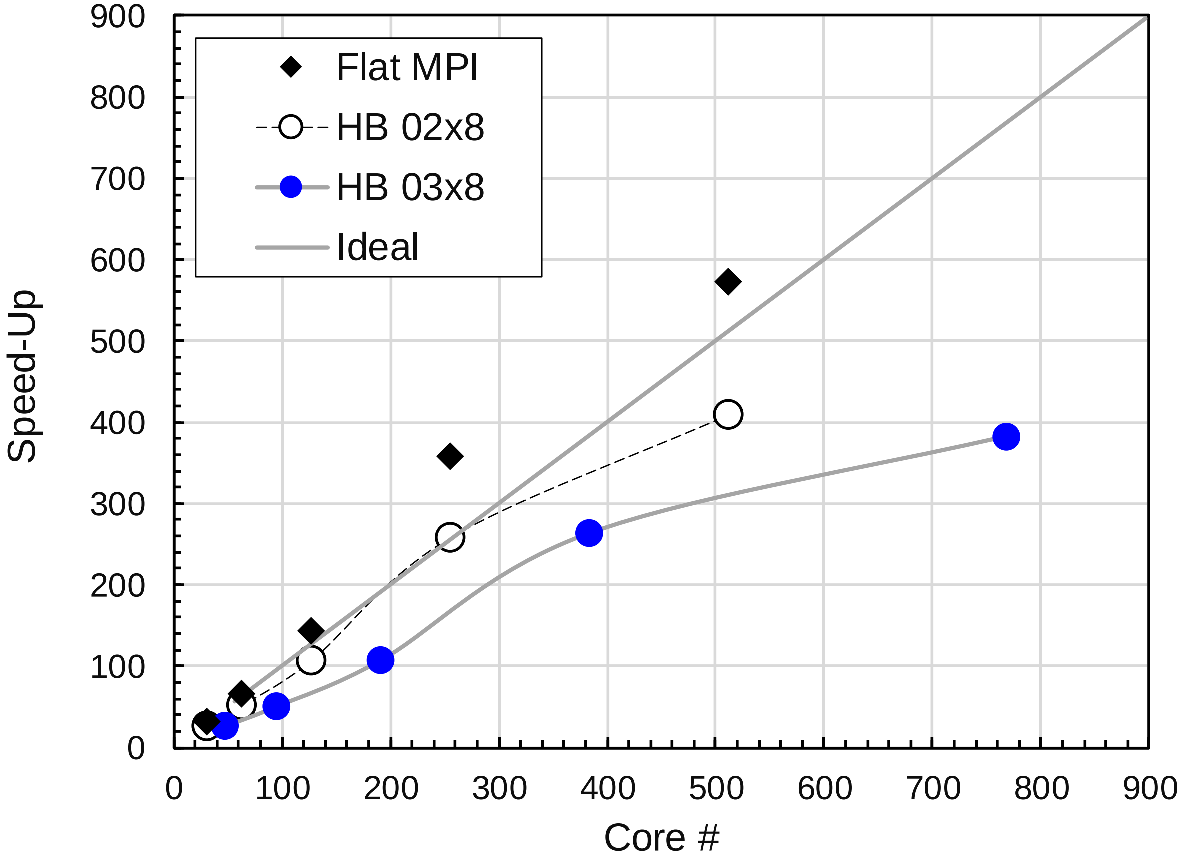


Example: Strong Scaling: Fortran

HB 2/3 x 8, 8-MPI Proc's on Soc.

- 128 × 128 × 128 nodes, 2,097,152 DOF
- 1-16 nodes, Linear Solver
- Best at 16-nodes





◆ Flat MPI
16 core's for each Socket
32 core's for each Node

○ HB 2x8
2x8= 16-cores/Soc,
32-cores/Node

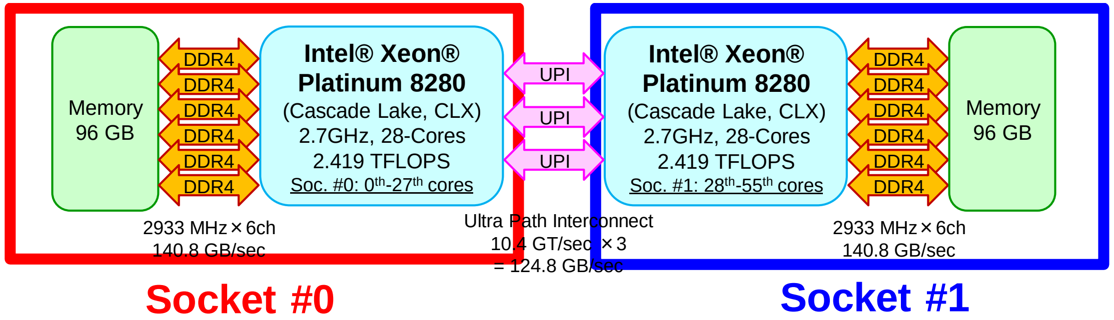
● HB 3x8
3x8= 24-cores/Soc
48-cores/Node

Performance of ◆ without NUMA at 32-cores= 32.0

HB (5or6or7) x 4, 8-nodes, 64-proc's

t05x04x2.sh t06x04x2.sh, t07x04x2.sh

mesh.inp
```
128 128 128
  4   4   4
pcube
```



t05x04x2.sh, HB 5x4
20/28 cores on ea. Soc.
5-threads x 4-proc's

```
#!/bin/sh
#PJM -N "HB05x04x2"
#PJM -L rscgrp=lecture9
#PJM -L node=8
#PJM --mpi proc=64
#PJM --omp thread=5
```

t06x04x2.sh, HB 6x4
24/28 cores on ea. Soc.
6-threads x 4-proc's

```
#!/bin/sh
#PJM -N "HB06x04x2"
#PJM -L rscgrp=lecture9
#PJM -L node=8
#PJM --mpi proc=64
#PJM --omp thread=6
```

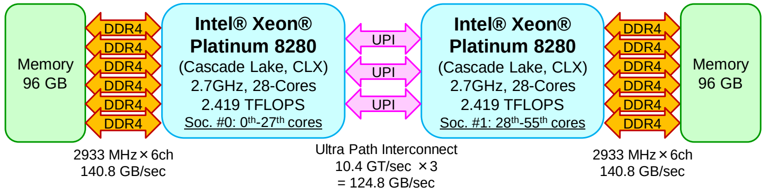
```
#!/bin/sh
#PJM -N "HB07x04x2"
#PJM -L rscgrp=lecture9
#PJM -L node=8
#PJM --mpi proc=64
#PJM --omp thread=7
```

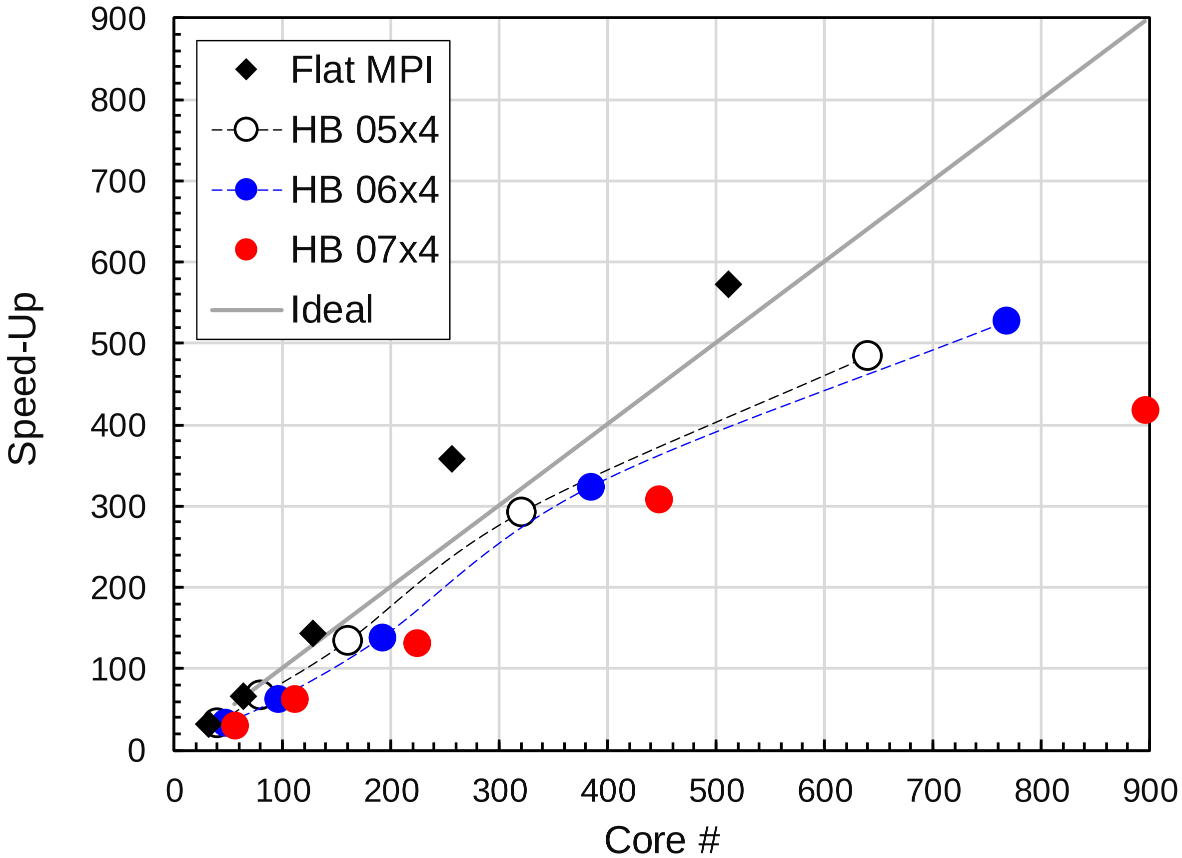
t07x04x2.sh, HB 7x4
28/28 cores on ea. Soc.
7-threads x 4-proc's

Example: Strong Scaling: Fortran

HB 5/6/7 x 4, 4-MPI Proc's on Soc.

- 128 × 128 × 128 nodes, 2,097,152 DOF
- 1-16 nodes, Linear Solver
- Best at 16-nodes





◆ Flat MPI
16 core's for each Socket
32 core's for each Node

○ HB 5x4
5x4= 20-cores/Socket
40-cores/Node

● HB 6x4
6x4= 24-cores/Socket
48-cores/Node

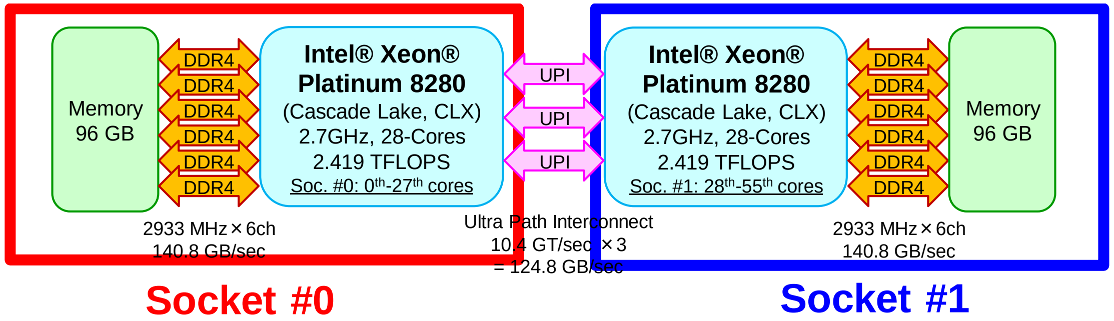
● HB 7x4
7x4= 28-cores/Socket
56-cores/Node

Performance of ◆ without NUMA at 32-cores= 32.0

HB (10/12/14) x 2, 8-nodes, 32-proc's

t10x02x2.sh t12x02x2.sh, t14x02x2.sh

```
mesh.inp
128 128 128
  4   4   2
pcube
```



t10x02x2.sh, HB 10x2
20/28 cores on ea. Soc.
10-threads x 2-proc's

```
#!/bin/sh
#PJM -N "HB10x02x2"
#PJM -L rscgrp=lecture9
#PJM -L node=8
#PJM --mpi proc=32
#PJM --omp thread=10
```

t12x02x2.sh, HB 12x2
24/28 cores on ea. Soc.
12-threads x 2-proc's

```
#!/bin/sh
#PJM -N "HB12x02x2"
#PJM -L rscgrp=lecture9
#PJM -L node=8
#PJM --mpi proc=32
#PJM --omp thread=12
```

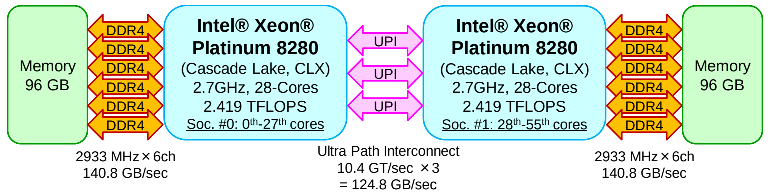
t14x02x2.sh, HB 14x2
28/28 cores on ea. Soc.
14-threads x 2-proc's

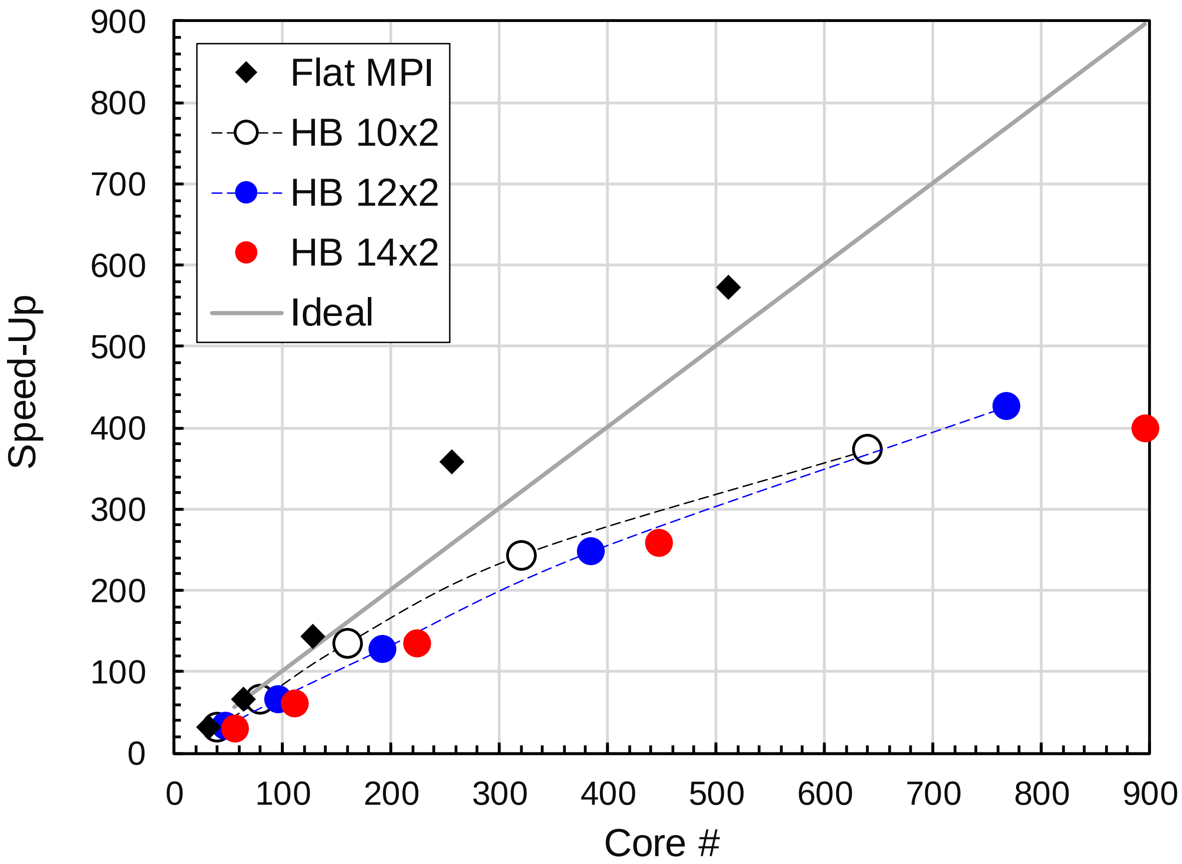
```
#!/bin/sh
#PJM -N "HB14x02x2"
#PJM -L rscgrp=lecture9
#PJM -L node=8
#PJM --mpi proc=32
#PJM --omp thread=14
```

Example: Strong Scaling: Fortran

HB 10/12/14 x 2, 2-MPI Proc's on Soc.

- 128×128×128 nodes, 2,097,152 DOF
- 1-16 nodes, Linear Solver
- Best at 16-nodes





◆ Flat MPI
16 core's for each Socket
32 core's for each Node

○ HB 10x2
10x2= 20-cores/Socket
40-cores/Node

● HB 12x2
12x2= 24-cores/Socket
48-cores/Node

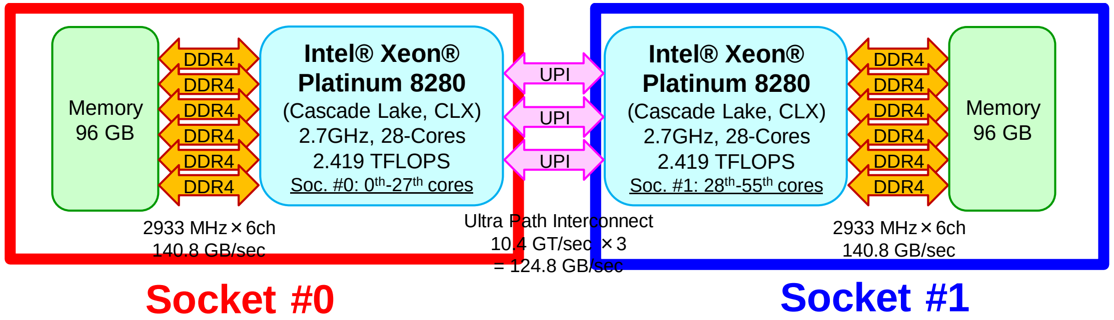
● HB 14x2
14x2= 28-cores/Socket
56-cores/Node

Performance of ◆ without NUMA at 32-cores= 32.0

HB (20/24/28) x 1, 8-nodes, 16-proc's

t20x01x2.sh t24x01x2.sh, t28x01x2.sh

```
mesh.inp
128 128 128
  4   2   2
pcube
```



t20x01x2.sh, HB 20x1
20/28 cores on ea. Soc.
20-threads x 1-proc's

```
#!/bin/sh
#PJM -N "HB20x01x2"
#PJM -L rscgrp=lecture9
#PJM -L node=8
#PJM --mpi proc=16
#PJM --omp thread=20
```

t24x01x2.sh, HB 24x1
24/28 cores on ea. Soc.
24-threads x 1-proc's

```
#!/bin/sh
#PJM -N "HB24x01x2"
#PJM -L rscgrp=lecture9
#PJM -L node=8
#PJM --mpi proc=16
#PJM --omp thread=24
```

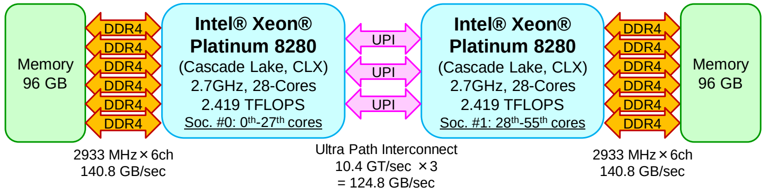
t28x01x2.sh, HB 28x1
28/28 cores on ea. Soc.
28-threads x 1-proc's

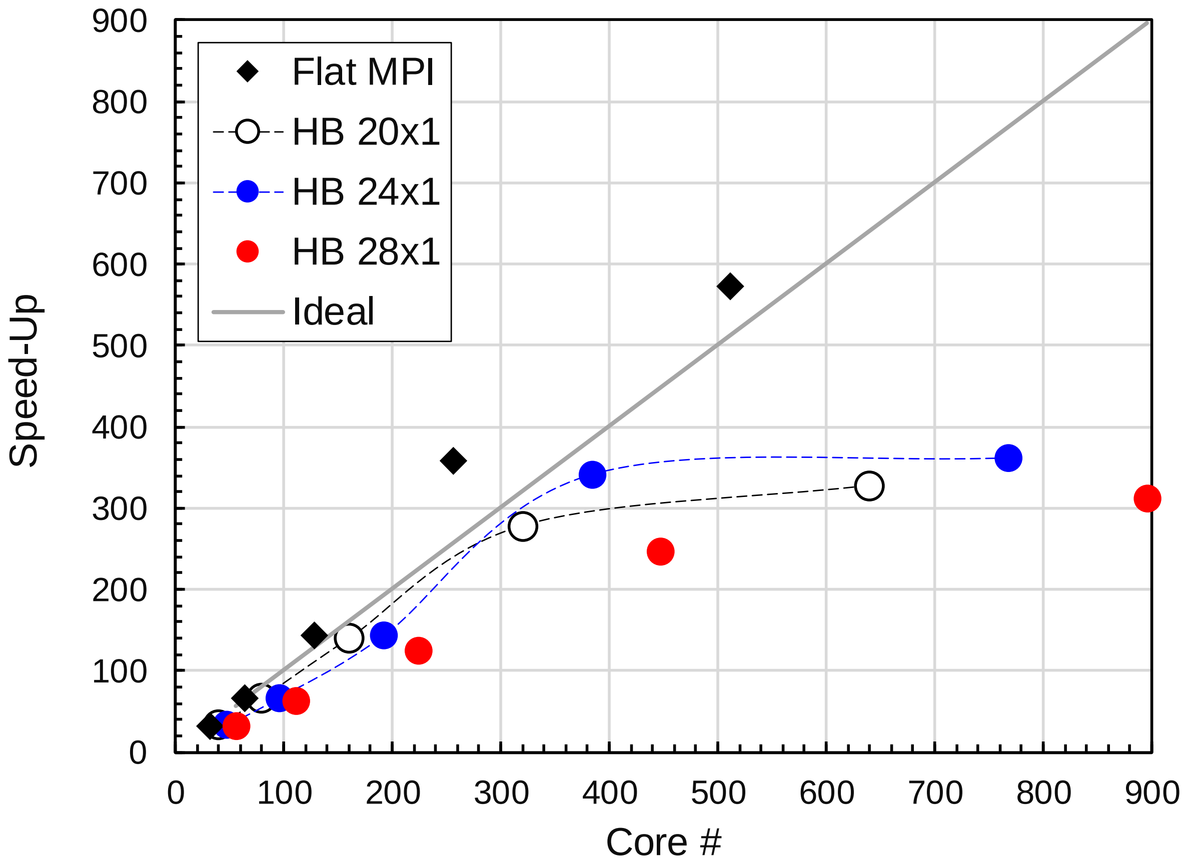
```
#!/bin/sh
#PJM -N "HB28x01x2"
#PJM -L rscgrp=lecture9
#PJM -L node=8
#PJM --mpi proc=16
#PJM --omp thread=28
```

Example: Strong Scaling: Fortran

HB 20/24/28 x 1, 1-MPI Proc's on Soc.

- 128 × 128 × 128 nodes, 2,097,152 DOF
- 1-16 nodes, Linear Solver
- Best at 16-nodes





◆ Flat MPI
16 core's for each Socket
32 core's for each Node

○ HB 20x1
20x1= 20-cores/Socket
40-cores/Node

● HB 24x1
24x1= 24-cores/Socket
48-cores/Node

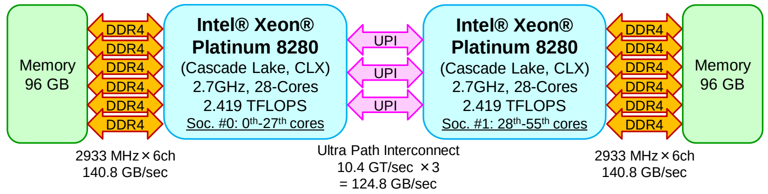
● HB 28x2
28x1= 28-cores/Socket
56-cores/Node

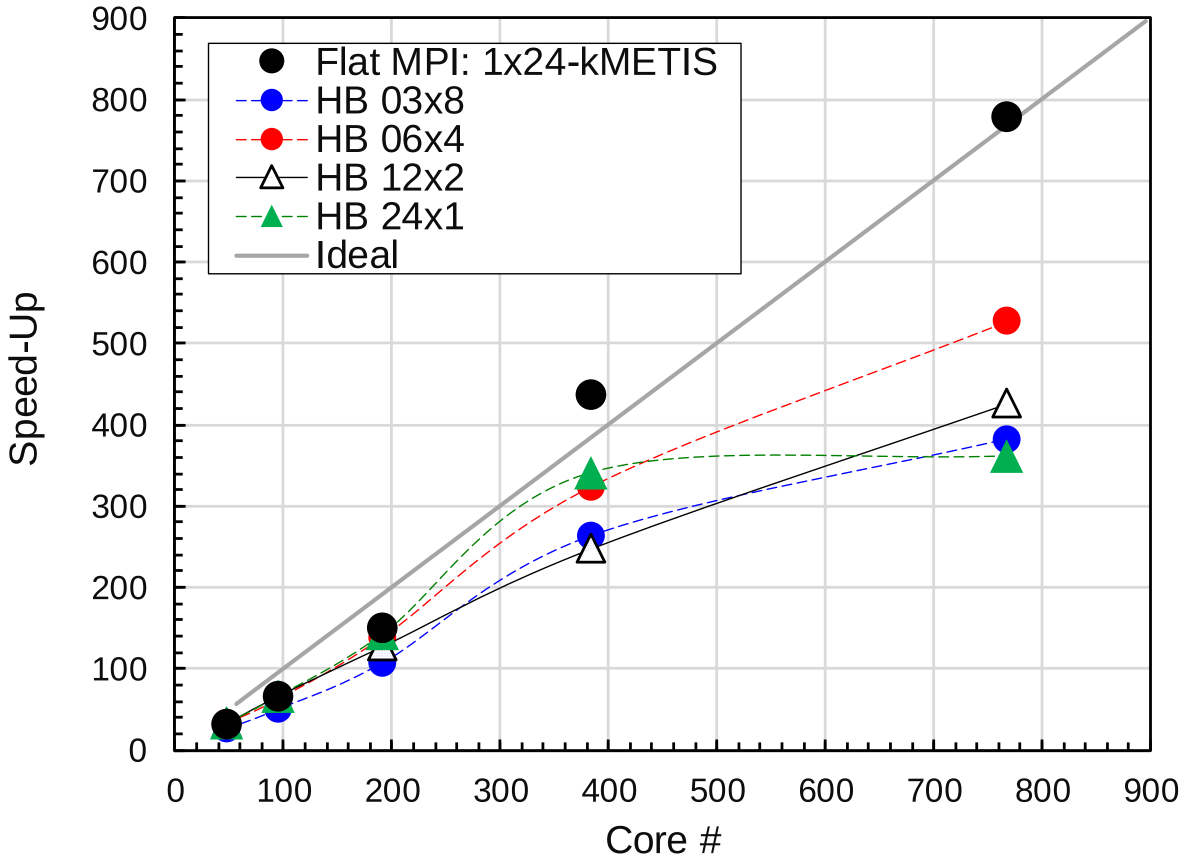
Performance of ◆ without NUMA at 32-cores= 32.0

Example: Strong Scaling: Fortran

Best Cases: 24-cores/Socket, 48-cores/Node

- 128 × 128 × 128 nodes, 2,097,152 DOF
- 1-16 nodes, Linear Solver
- Best at 16-nodes





- Flat MPI 1x24 k-METIS
  24 core's for each Socket
  48 core's for each Node

- HB 3x8
- HB 6x4
- HB 12x2
- HB 24x1

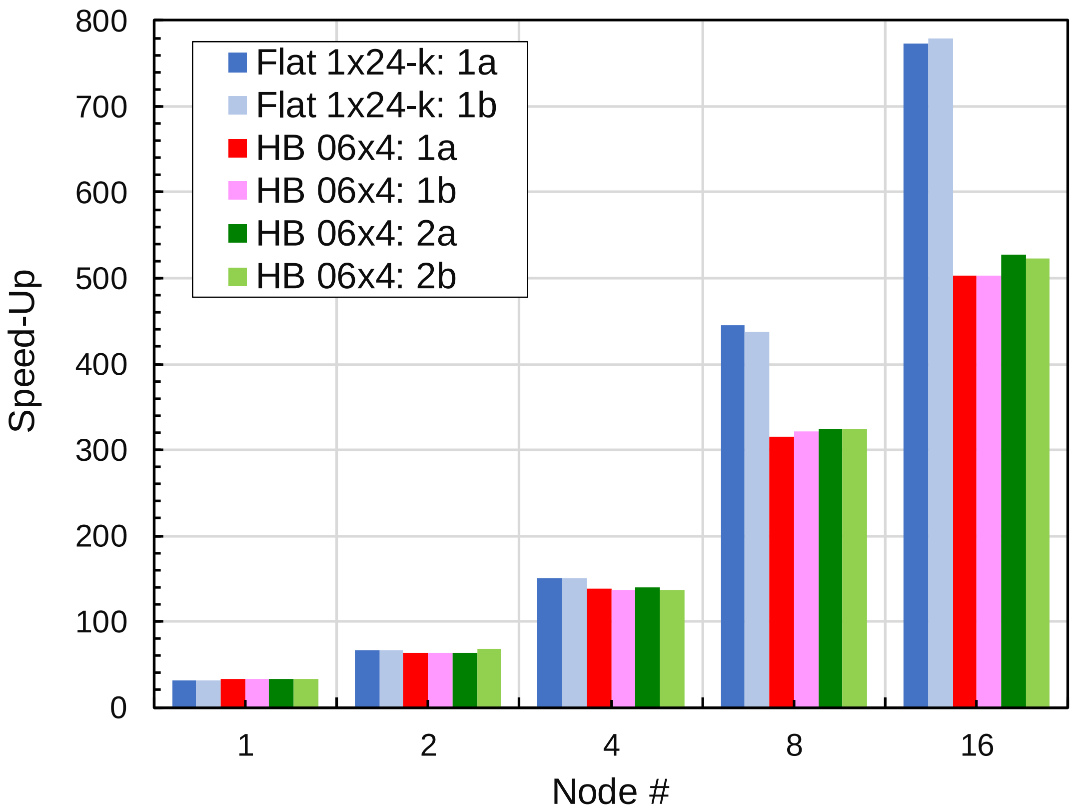
Performance of ◆ (Flat MPI 1x16) without NUMA at 32-cores= 32.0

Example: Strong Scaling: Fortran

Best Cases: 24-cores/Socket, 48-cores/Node

- 1a: No Options
- 1b: numactl –l
- 2a: KMP_AFFINITY
- 2b: KMP_AFFINITY + numactl -l

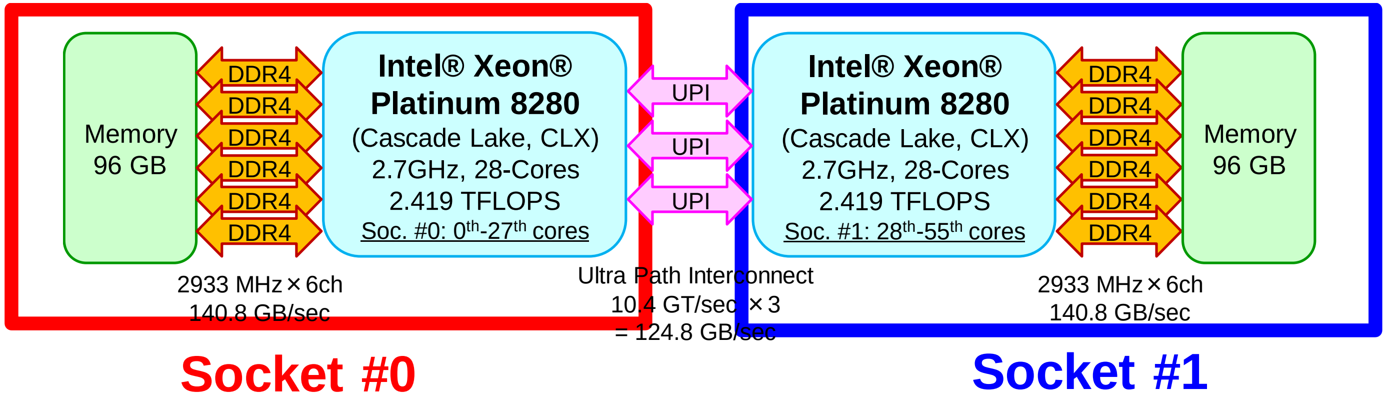
```
mpiexec.hydra -n ${PJM_MPI_PROC} ./sol1                1a
mpiexec.hydra -n ${PJM_MPI_PROC} numactl -l ./sol1     1b

export KMP_AFFINITY=granularity=fine,compact
mpiexec.hydra -n ${PJM_MPI_PROC} ./sol1                2a
mpiexec.hydra -n ${PJM_MPI_PROC} numactl -l ./sol1     2b
```



Performance of ◆ (Flat MPI 1x16) without NUMA at 32-cores= 32.0

Flat MPI vs. Hybrid

- アプリ・HWの特性，問題サイズ等に依存
- 一般的にノード数が少ない場合には疎行列ソルバーを含むアプリではFlat MPIの性能が良い
  - メモリ性能
- ノード数が増えるとHybridが良くなると言われるが
  - MPIプロセス数は少なくなる
  - OBCXにおけるHBの挙動はやや不可解
- NUMAに配慮すれば，各ノード当たりに1MPIプロセスとすることも可能

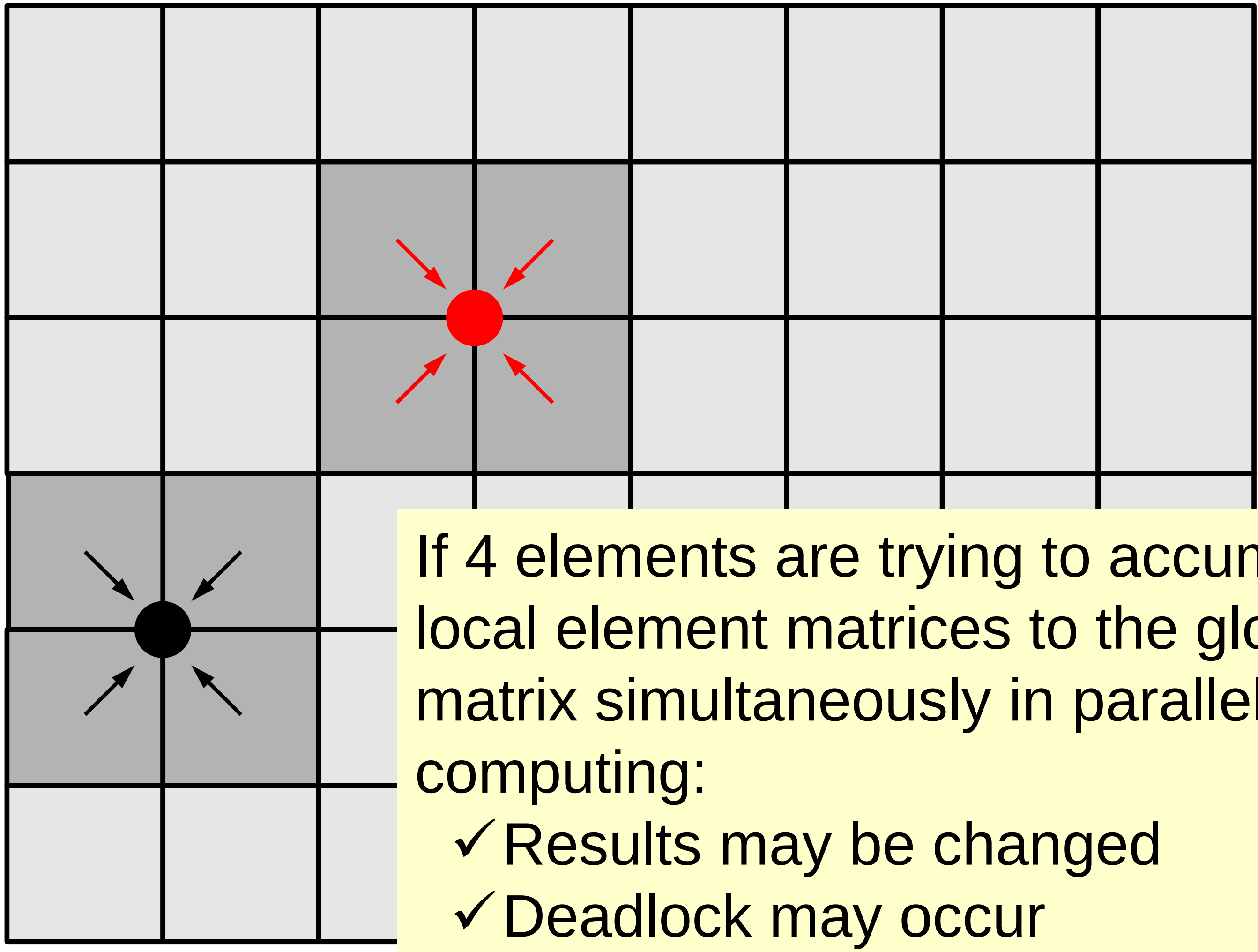


How to apply multi-threading

- CG Solver
  - OpenMP指示文を挿入するのみ
  - ILU/IC 前処理の場合はもっと難しい
- MAT_ASS (mat_ass_main, mat_ass_bc)
  - データ依存性あり
  - 複数要素による1節点への足し込みが並列計算時に同時に発生することを避ける必要がある
    - 答えが変わる，もしくはDead Lockが生じる可能性がある
  - 色づけ：Coloring
    - 同じ色に彩色された要素は節点を共有しない
    - 同じ色の要素には並列計算が可能
    - 本問題の場合，三次元では8色，二次元では4色必要
    - 色づけ部分の計算はexpensive：並列化困難

Mat_Ass: Data Dependency

Each Node is shared by 4-Elements in 2D



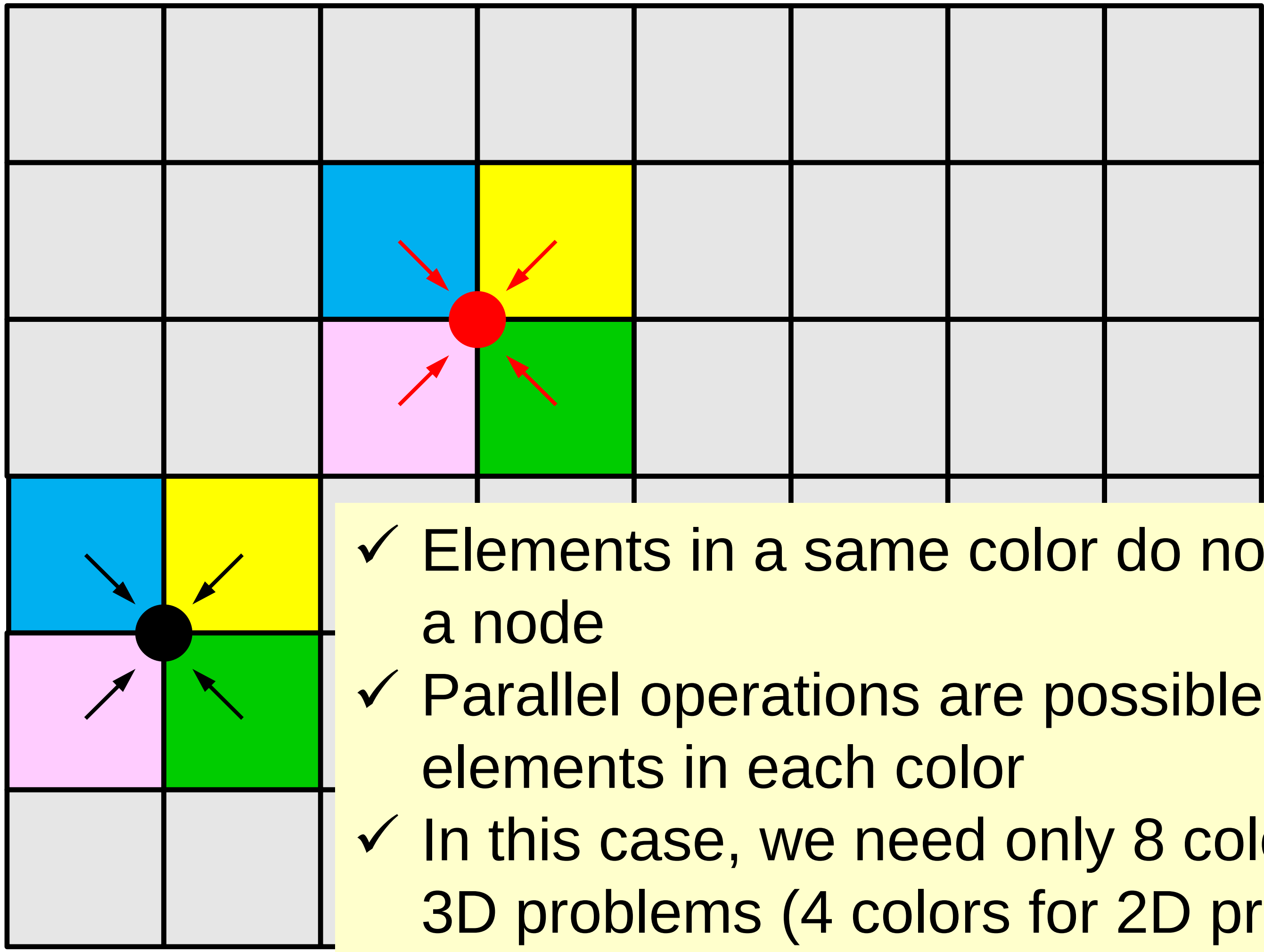
If 4 elements are trying to accumulate local element matrices to the global matrix simultaneously in parallel computing:

- ✓Results may be changed
- ✓Deadlock may occur

How to apply multi-threading

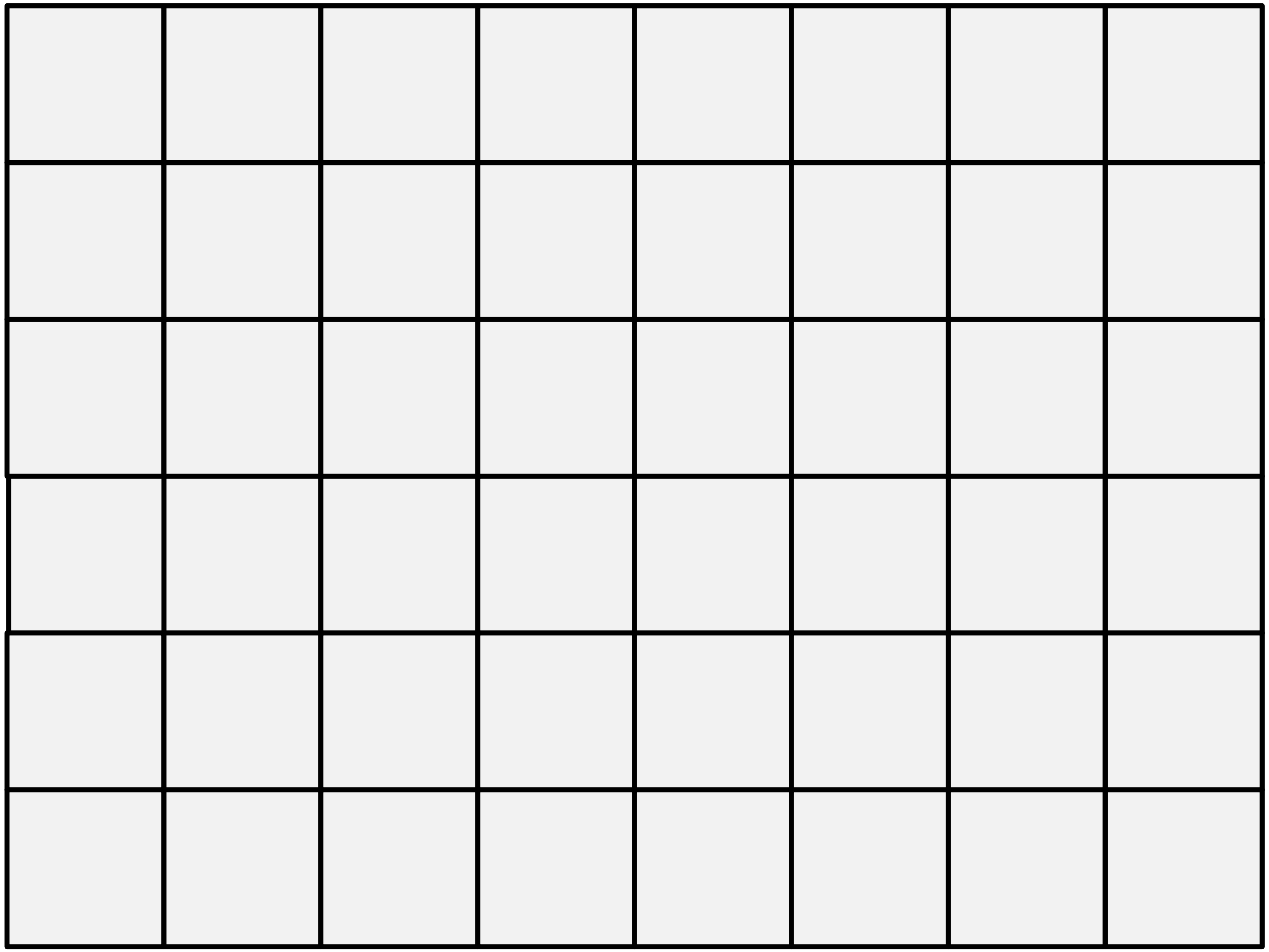
- CG Solver
  - OpenMP指示文を挿入するのみ
  - ILU/IC 前処理の場合はもっと難しい
- MAT_ASS (mat_ass_main, mat_ass_bc)
  - 色づけ：Coloring
    - 同じ色に彩色された要素は節点を共有しない
    - 同じ色の要素には並列計算が可能
    - 本問題の場合，三次元では8色，二次元では4色必要

Mat_Ass: Data Dependency



- ✓ Elements in a same color do not share a node
- ✓ Parallel operations are possible for elements in each color
- ✓ In this case, we need only 8 colors for 3D problems (4 colors for 2D problems)

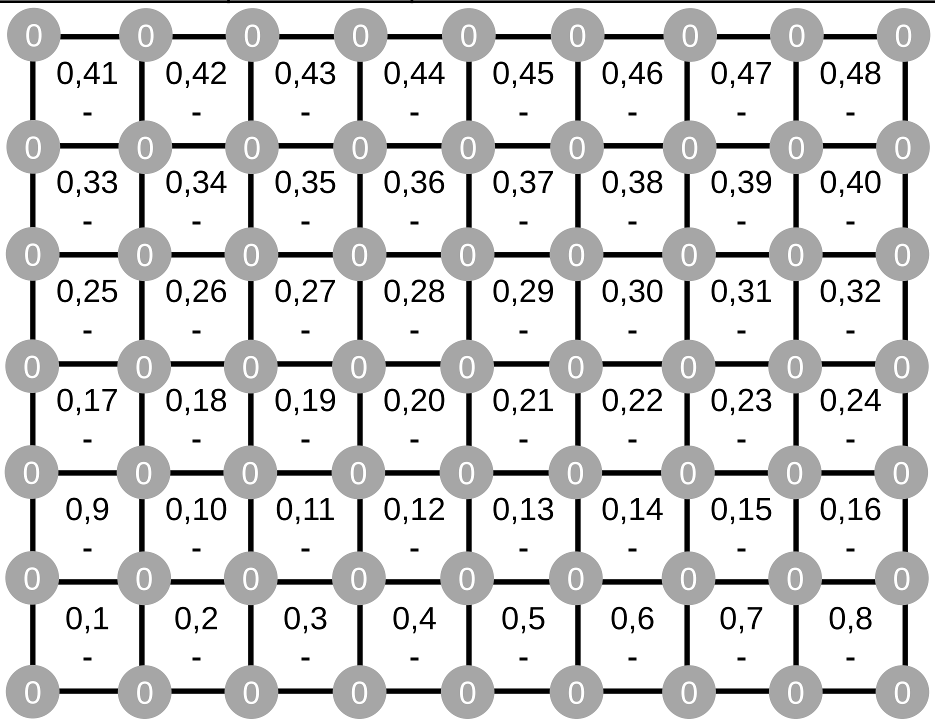
Target: 8x6= 48-meshes



Coloring (2D) (1/7)

```
      allocate (ELMCOLORindex(0:NP))

      allocate (ELMCOLORitem (ICELTOT))
      allocate (W1(NP), W2(ICELTOT), W3(NP))

      W1=0; W2=0; W3=0
      icou= 0
      do icol= 1, NP
        do i= 1, NP
          W1(i)= 0
        enddo
        do icel= 1, ICELTOT
          if (W2(icel).eq.0) then
            in1= ICELNOD(icel,1)
            in2= ICELNOD(icel,2)
            in3= ICELNOD(icel,3)
            in4= ICELNOD(icel,4)

            ip1= W1(in1)
            ip2= W1(in2)
            ip3= W1(in3)
            ip4= W1(in4)

            isum= ip1 + ip2 + ip3 + ip4
            if (isum.eq.0) then
              icou= icou + 1
              W3(icol)= icou
              W2(icel)= icol

              ELMCOLORitem(icou)= icel
              W1(in1)= 1
              W1(in2)= 1
              W1(in3)= 1
              W1(in4)= 1
            endif
            if (icou.eq.ICELTOT) goto 100
          endif
        enddo
      enddo

 100  continue
      ELMCOLORtot= icol
      W3(0          )= 0
      W3(ELMCOLORtot)= ICELTOT

      do icol= 0, ELMCOLORtot
        ELMCOLORindex(icol)= W3(icol)
      enddo
```

Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 0 1	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 0	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors



Coloring (2D) (1/7)

```
      allocate (ELMCOLORindex(0:NP))

      allocate (ELMCOLORitem (ICELTOT))
      allocate (W1(NP), W2(ICELTOT), W3(NP))

      W1=0; W2=0; W3=0
      icou= 0
      do icol= 1, NP
        do i= 1, NP
          W1(i)= 0
        enddo
        do icel= 1, ICELTOT
```

W2(icel), IDold IDnew

W2: Color ID of the Element
IDold: Element ID (Original)
IDnew: Element ID (New)

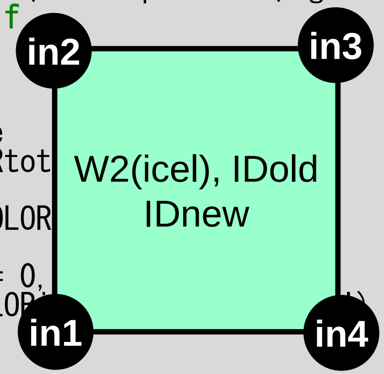
```
      enddo

100   continue
      ELMCOLORtot= icol
      W3(0          )= 0
      W3(ELMCOLORtot)= ICELTOT

      do icol= 0, ELMCOLORtot
        ELMCOLORindex(icol)= W3(icol)
      enddo
```

Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 (0) (1)	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 [0]	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors

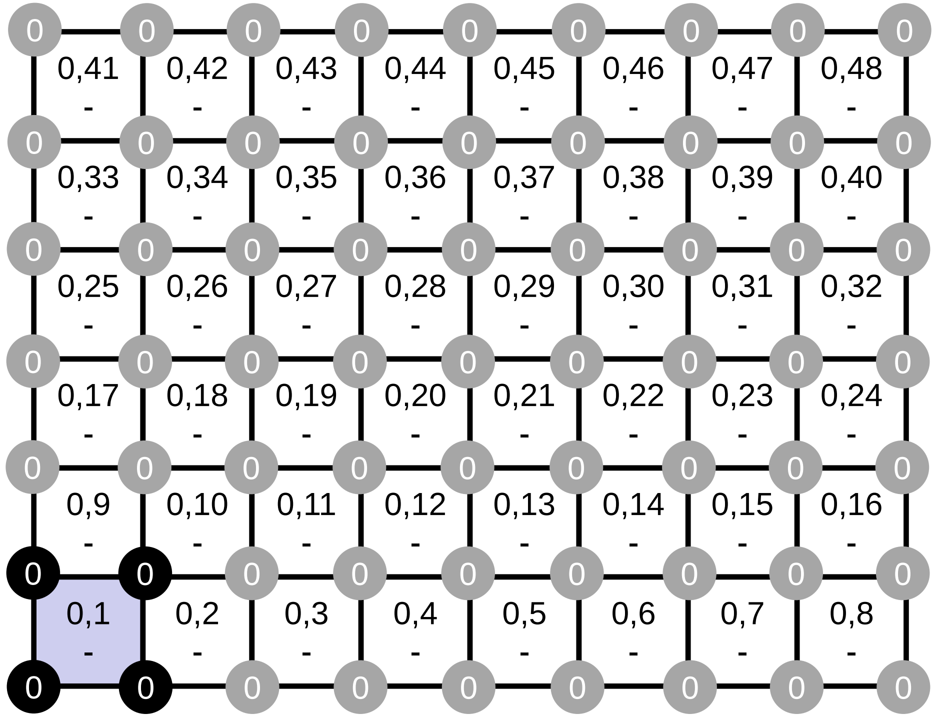
Coloring (2D) (2/7)

```
      allocate (ELMCOLORindex(0:NP))

      allocate (ELMCOLORitem (ICELTOT))
      allocate (W1(NP), W2(ICELTOT), W3(NP))

      W1=0; W2=0; W3=0
      icou= 0
      do icol= 1, NP
        do i= 1, NP
          W1(i)= 0
        enddo
        do icel= 1, ICELTOT
          if (W2(icel).eq.0) then
            in1= ICELNOD(icel,1)
            in2= ICELNOD(icel,2)
            in3= ICELNOD(icel,3)
            in4= ICELNOD(icel,4)

            ip1= W1(in1)  (=0)
            ip2= W1(in2)  (=0)
            ip3= W1(in3)  (=0)
            ip4= W1(in4)  (=0)

            isum= ip1 + ip2 + ip3 + ip4
            if (isum.eq.0) then
              icou= icou + 1
              W3(icol)= icou
              W2(icel)= icol

              ELMCOLORitem(icou)= icel
              W1(in1)= 1
              W1(in2)= 1
              W1(in3)= 1
              W1(in4)= 1
            endif
            if (icou.eq.ICELTOT) goto 100
          endif
        enddo
      enddo

 100  continue
      ELMCOLORtot
      W3(0
      W3(ELMCOLOR

      do icol= 0,
        ELMCOLOR
      enddo
```

icol=1
icel=1



Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 0 1	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 0	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors

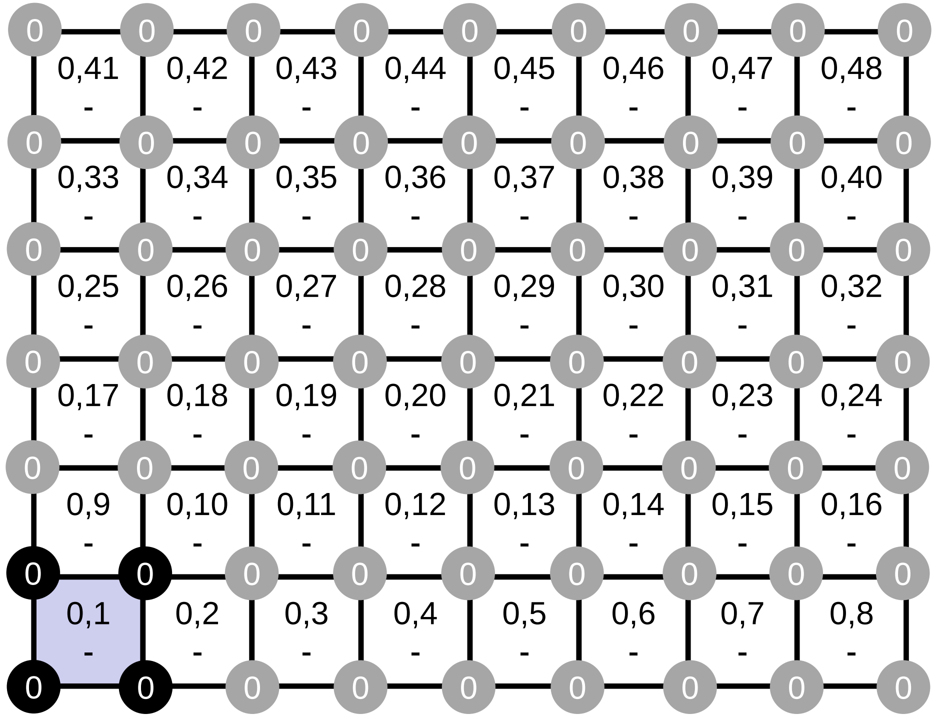


Coloring (2D) (2/7)

```
      allocate (ELMCOLORindex(0:NP))

      allocate (ELMCOLORitem (ICELTOT))
      allocate (W1(NP), W2(ICELTOT), W3(NP))

      W1=0; W2=0; W3=0
      icou= 0
      do icol= 1, NP
        do i= 1, NP
          W1(i)= 0
        enddo
        do icel= 1, ICELTOT
          if (W2(icel).eq.0) then
            in1= ICELNOD(icel,1)
            in2= ICELNOD(icel,2)
            in3= ICELNOD(icel,3)
            in4= ICELNOD(icel,4)

            ip1= W1(in1)  (=0)
            ip2= W1(in2)  (=0)
            ip3= W1(in3)  (=0)
            ip4= W1(in4)  (=0)

            isum= ip1 + ip2 + ip3 + ip4  (=0)
            if (isum.eq.0) then
              icou= icou + 1
              W3(icol)= icou
              W2(icel)= icol

              ELMCOLORitem(icou)= icel
              W1(in1)= 1
              W1(in2)= 1
              W1(in3)= 1
              W1(in4)= 1
            endif
            if (icou.eq.ICELTOT) goto 100
          endif
        enddo
      enddo

 100  continue
      ELMCOLORtot= icol
      IWKX(0          ,3)= 0
      IWKX(ELMCOLORtot,3)= ICELTOT

      do icol= 0, ELMCOLORtot
        ELMCOLORindex(icol)= W3(icol)
      enddo
```

icol=1
icel=1

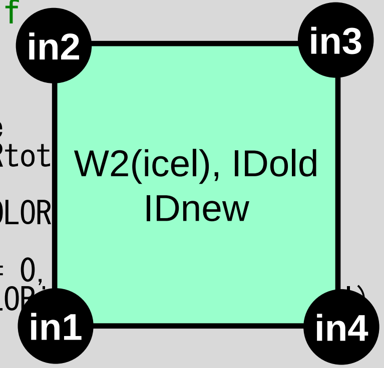
Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 0 1	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 0	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors



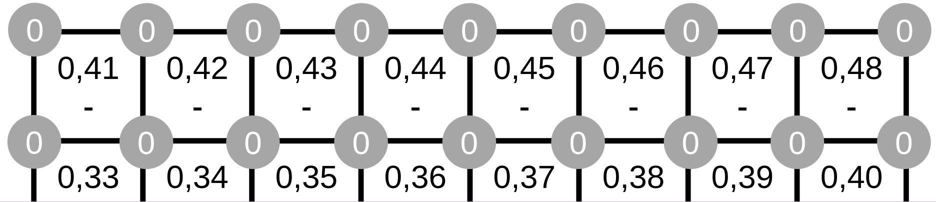
Coloring (2D) (2/7)

```
      allocate (ELMCOLORindex(0:NP))

      allocate (ELMCOLORitem (ICELTOT))
      allocate (W1(NP), W2(ICELTOT), W3(NP))

      W1=0; W2=0; W3=0
      icou= 0
      do icol= 1, NP
        do i= 1, NP
          W1(i)= 0
        enddo
        do icel= 1, ICELTOT
          if (W2(icel).eq.0) then
            in1= ICELNOD(icel,1)
            in2= ICELNOD(icel,2)
            in3= ICELNOD(icel,3)
            in4= ICELNOD(icel,4)

            ip1= W1(in1)  (=0)
            ip2= W1(in2)  (=0)
            ip3= W1(in3)  (=0)
            ip4= W1(in4)  (=0)

            isum= ip1 + ip2 + ip3 + ip4  (=0)
            if (isum.eq.0) then
              icou= icou + 1
              W3(icol)= icou
              W2(icel)= icol

              ELMCOLORitem(icou)= icel
              W1(in1)= 1
              W1(in2)= 1
              W1(in3)= 1
              W1(in4)= 1
            endif
            if (icou.eq.ICELTOT) goto
          endif
        enddo
      enddo

 100  continue
      ELMCOLORtot
      W3(0
      W3(ELMCOLOR

      do icol= 0,
        ELMCOLOR
      enddo
```

icol=1
icel=1

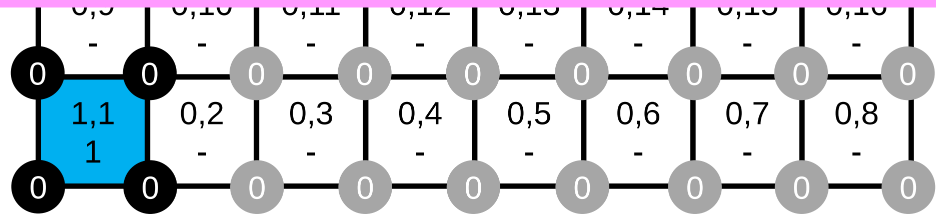


Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 0 1	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 0	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors



Because no vertices on this element were "flagged" yet in this Color (=icol), this element can join this Color (=icol) !!

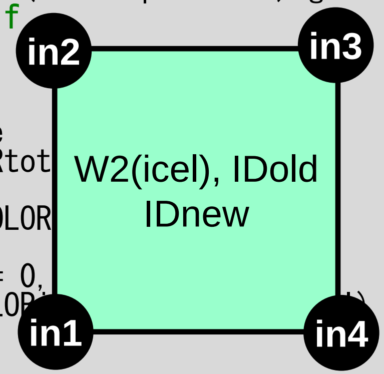
icou= icou + 1 Colored Element #, NEW Element ID
W3(icol)= icou Accumulated # of Colored Elem's in Each Color
W2(icel)= icol Color ID of Each Element
ELMCOLORitem(icou)= icel OLD Element ID



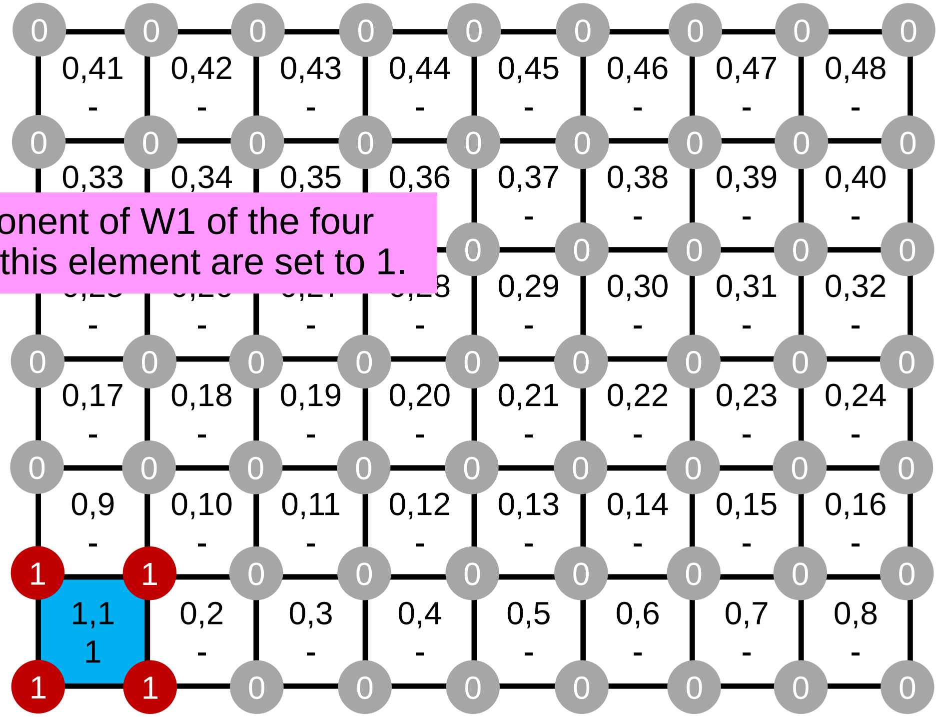
Coloring (2D) (2/7)

```
      allocate (ELMCOLORindex(0:NP))

      allocate (ELMCOLORitem (ICELTOT))
      allocate (W1(NP), W2(ICELTOT), W3(NP))

      W1=0; W2=0; W3=0
      icou= 0
      do icol= 1, NP
        do i= 1, NP
          W1(i)= 0
        enddo
        do icel= 1, ICELTOT
          if (W2(icel).eq.0) then
            in1= ICELNOD(icel,1)
            in2= ICELNOD(icel,2)
            in3= ICELNOD(icel,3)
            in4= ICELNOD(icel,4)

            ip1= W1(in1)  (=0)
            ip2= W1(in2)  (=0)
            ip3= W1(in3)  (=0)
            ip4= W1(in4)  (=0)

            isum= ip1 + ip2 + ip3 + ip4  (=0)
            if (isum.eq.0) then
              icou= icou + 1
              W3(icol)= icou
              W2(icel)= icol

              ELMCOLORitem(icou)= icel
              W1(in1)= 1
              W1(in2)= 1
              W1(in3)= 1
              W1(in4)= 1
            endif
            if (icou.eq.ICELTOT) goto 100
          endif
        enddo
      enddo

 100  continue
      ELMCOLORtot
      W3(0
      W3(ELMCOLOR

      do icol= 0,
        ELMCOLOR
      enddo
```

icol=1
icel=1



Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 (0) (1)	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 (0)	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors



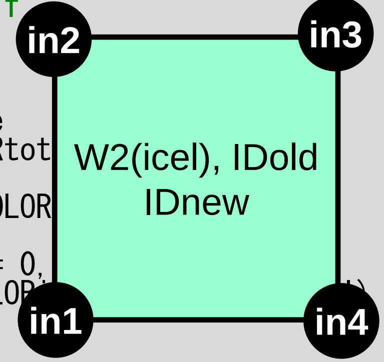
Each component of W1 of the four vertices on this element are set to 1.

Coloring (2D) (3/7)

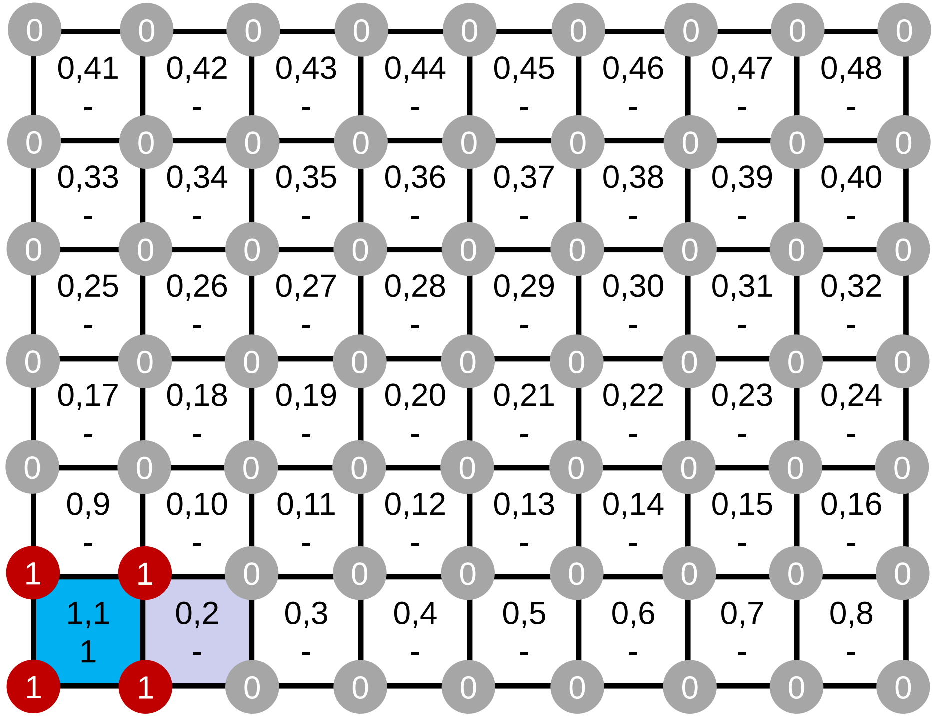
```
allocate (ELMCOLORindex(0:NP))

allocate (ELMCOLORitem (ICELTOT))
allocate (W1(NP), W2(ICELTOT), W3(NP))

W1=0; W2=0; W3=0
icou= 0
do icol= 1, NP
  do i= 1, NP
    W1(i)= 0
  enddo
  do icel= 1, ICELTOT
    if (W2(icel).eq.0) then
      in1= ICELNOD(icel,1)
      in2= ICELNOD(icel,2)
      in3= ICELNOD(icel,3)
      in4= ICELNOD(icel,4)

      ip1= W1(in1)  (=1)
      ip2= W1(in2)  (=1)
      ip3= W1(in3)  (=0)
      ip4= W1(in4)  (=0)

      isum= ip1 + ip2 + ip3 + ip4  (=2)
      if (isum.eq.0) then
        icou= icou + 1
        W3(icol)= icou

    endif
  enddo
enddo

100 continue
    ELMCOLORtot
    W3(0
    W3(ELMCOLOR

    do icol= 0,
      ELMCOLOR
    enddo
```

icol=1
icel=2

- ✓ 2 of 4 vertices on this element are already "flagged" (isum=2)
- ✓ Elements in a same color do not share a node
- ✓ Therefore, this element (icel=2) cannot join this color (icol=1)



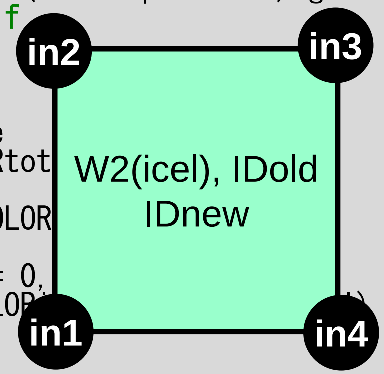
Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 (0) (1)	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 [0]	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors



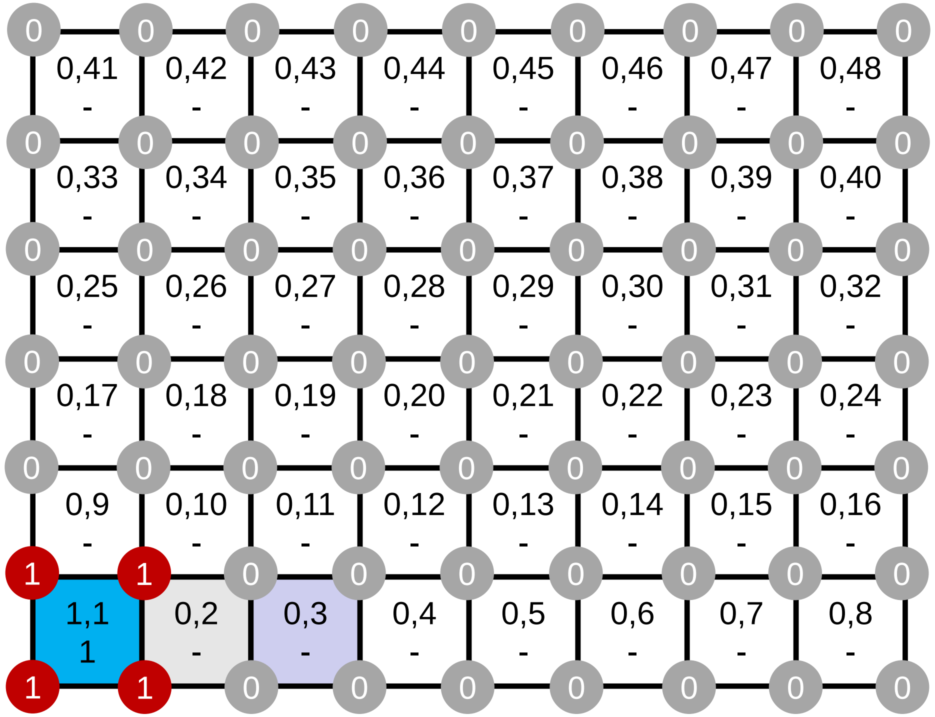
Coloring (2D) (4/7)

```
allocate (ELMCOLORindex(0:NP))

allocate (ELMCOLORitem (ICELTOT))
allocate (W1(NP), W2(ICELTOT), W3(NP))

W1=0; W2=0; W3=0
icou= 0
do icol= 1, NP
  do i= 1, NP
    W1(i)= 0
  enddo
  do icel= 1, ICELTOT
    if (W2(icel).eq.0) then
      in1= ICELNOD(icel,1)
      in2= ICELNOD(icel,2)
      in3= ICELNOD(icel,3)
      in4= ICELNOD(icel,4)

      ip1= W1(in1)  (=0)
      ip2= W1(in2)  (=0)
      ip3= W1(in3)  (=0)
      ip4= W1(in4)  (=0)

      isum= ip1 + ip2 + ip3 + ip4  (=0)
      if (isum.eq.0) then
        icou= icou + 1
        W3(icol)= icou
        W2(icel)= icol

        ELMCOLORitem(icou)= icel
        W1(in1)= 1
        W1(in2)= 1
        W1(in3)= 1
        W1(in4)= 1
      endif
      if (icou.eq.ICELTOT) goto 100
    endif
  enddo
enddo

100 continue
    ELMCOLORtot
    W3(0
    W3(ELMCOLOR

    do icol= 0,
      ELMCOLOR
    enddo
```

icol=1
icel=3



Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 0 1	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 0	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors

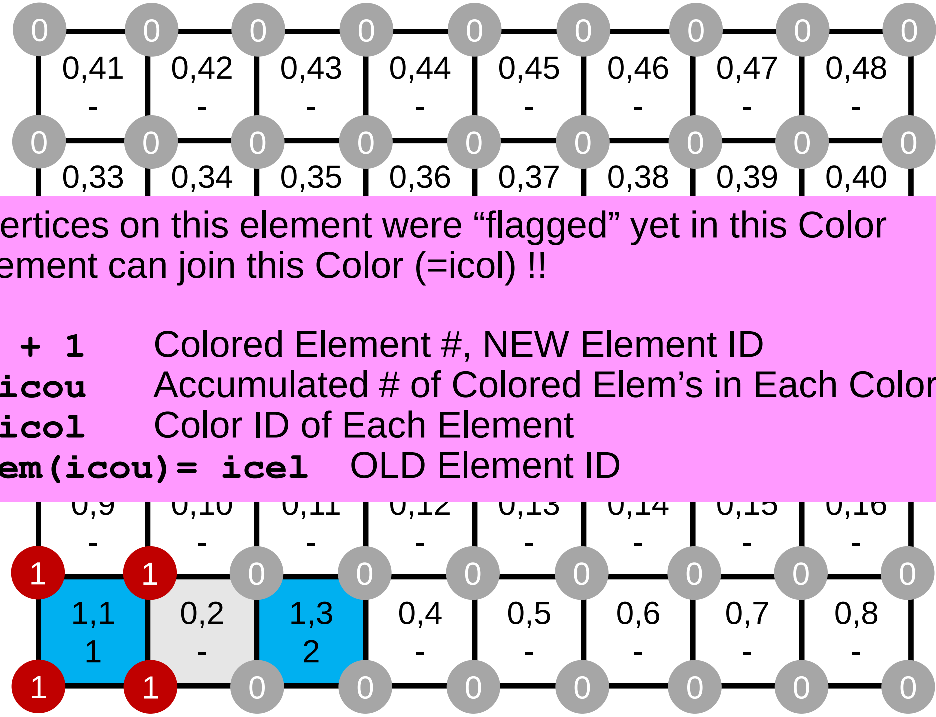


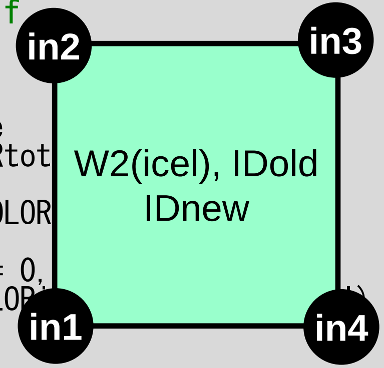
Coloring (2D) (4/7)

```
      allocate (ELMCOLORindex(0:NP))

      allocate (ELMCOLORitem (ICELTOT))
      allocate (W1(NP), W2(ICELTOT), W3(NP))

      W1=0; W2=0; W3=0
      icou= 0
      do icol= 1, NP
        do i= 1, NP
          W1(i)= 0
        enddo
        do icel= 1, ICELTOT
          if (W2(icel).eq.0) then
            in1= ICELNOD(icel,1)
            in2= ICELNOD(icel,2)
            in3= ICELNOD(icel,3)
            in4= ICELNOD(icel,4)

            ip1= W1(in1)  (=0)
            ip2= W1(in2)  (=0)
            ip3= W1(in3)  (=0)
            ip4= W1(in4)  (=0)

            isum= ip1 + ip2 + ip3 + ip4  (=0)
            if (isum.eq.0) then
              icou= icou + 1
              W3(icol)= icou
              W2(icel)= icol

              ELMCOLORitem(icou)= icel
              W1(in1)= 1
              W1(in2)= 1
              W1(in3)= 1
              W1(in4)= 1
            endif
            if (icou.eq.ICELTOT) goto
          endif
        enddo
      enddo

 100  continue
      ELMCOLORtot
      W3(0
      W3(ELMCOLOR

      do icol= 0,
        ELMCOLOR
      enddo
```

icol=1
icel=3

Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 0 1	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 0	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors





Because no vertices on this element were "flagged" yet in this Color (=icol), this element can join this Color (=icol) !!

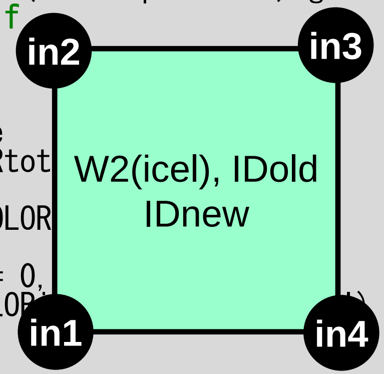
icou= icou + 1	Colored Element #, NEW Element ID
W3(icol)= icou	Accumulated # of Colored Elem's in Each Color
W2(icel)= icol	Color ID of Each Element
ELMCOLORitem(icou)= icel	OLD Element ID

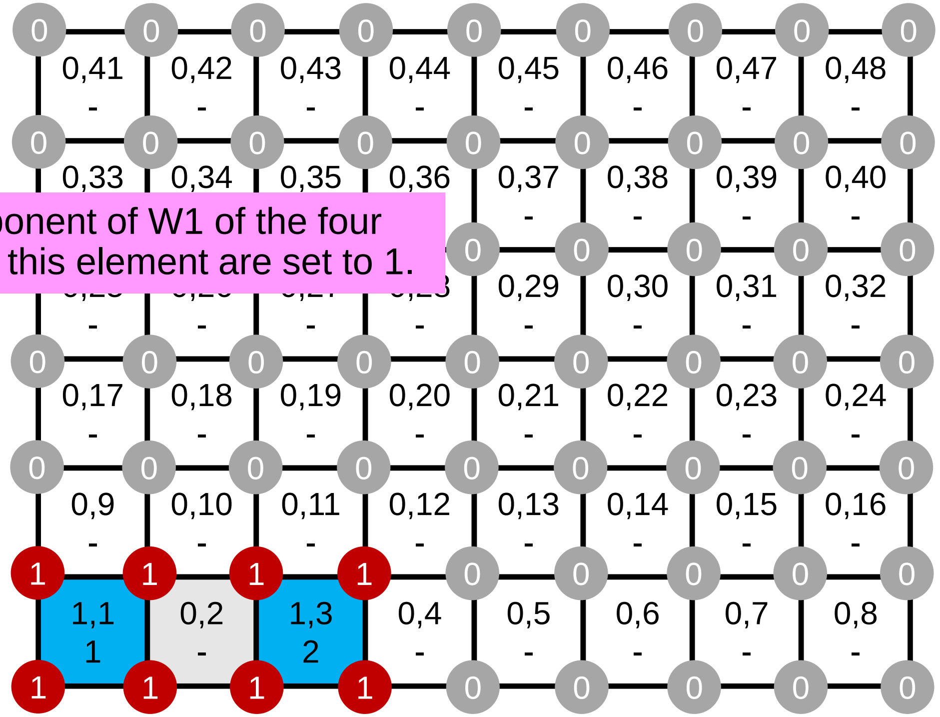
Coloring (2D) (4/7)

```
      allocate (ELMCOLORindex(0:NP))

      allocate (ELMCOLORitem (ICELTOT))
      allocate (W1(NP), W2(ICELTOT), W3(NP))

      W1=0; W2=0; W3=0
      icou= 0
      do icol= 1, NP
        do i= 1, NP
          W1(i)= 0
        enddo
        do icel= 1, ICELTOT
          if (W2(icel).eq.0) then
            in1= ICELNOD(icel,1)
            in2= ICELNOD(icel,2)
            in3= ICELNOD(icel,3)
            in4= ICELNOD(icel,4)

            ip1= W1(in1)  (=0)
            ip2= W1(in2)  (=0)
            ip3= W1(in3)  (=0)
            ip4= W1(in4)  (=0)

            isum= ip1 + ip2 + ip3 + ip4  (=0)
            if (isum.eq.0) then
              icou= icou + 1
              W3(icol)= icou
              W2(icel)= icol

              ELMCOLORitem(icou)= icel
              W1(in1)= 1
              W1(in2)= 1
              W1(in3)= 1
              W1(in4)= 1
            endif
            if (icou.eq.ICELTOT) goto 100
          endif
        enddo
      enddo

 100  continue
      ELMCOLORtot
      W3(0
      W3(ELMCOLOR

      do icol= 0,
        ELMCOLOR
      enddo
```

icol=1
icel=3

Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 0 1	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 0	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors



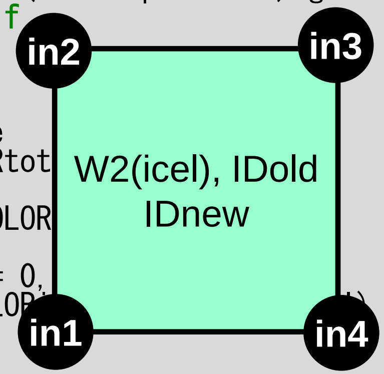


Each component of W1 of the four vertices on this element are set to 1.

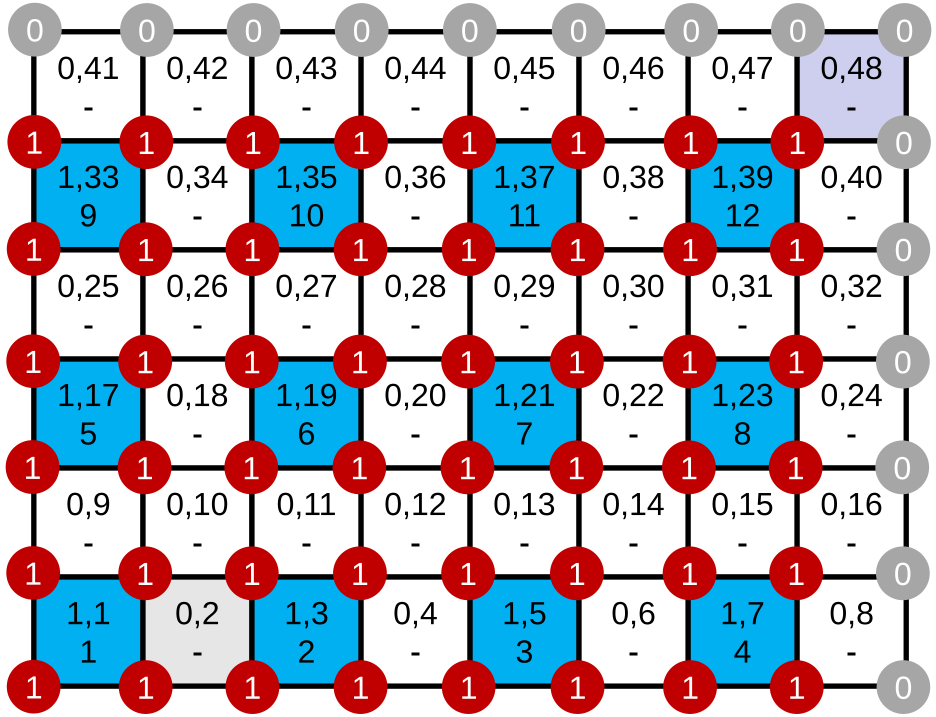
Coloring (2D) (5/7)

icol=1
icel=ICELTOT (=48)

```
      allocate (ELMCOLORindex(0:NP))

      allocate (E
      allocate (W

      W1=0; W2=0;
      icou= 0
      do icol= 1, NP
        do i= 1, NP
          W1(i)= 0
        enddo
        do icel= 1, ICELTOT
          if (W2(icel).eq.0) then
            in1= ICELNOD(icel,1)
            in2= ICELNOD(icel,2)
            in3= ICELNOD(icel,3)
            in4= ICELNOD(icel,4)

            ip1= W1(in1)  (=1)
            ip2= W1(in2)  (=0)
            ip3= W1(in3)  (=0)
            ip4= W1(in4)  (=0)

            isum= ip1 + ip2 + ip3 + ip4  (=1)
            if (isum.eq.0) then
              icou= icou + 1
              W3(icol)= icou
              W2(icel)= icol

              ELMCOLORitem(icou)= icel
              W1(in1)= 1
              W1(in2)= 1
              W1(in3)= 1
              W1(in4)= 1
            endif
            if (icou.eq.ICELTOT) goto 100
          endif
        enddo
      enddo

 100  continue
      ELMCOLORtot
      W3(0
      W3(ELMCOLOR

      do icol= 0,
        ELMCOLOR
      enddo
```



Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 (0) (1)	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 [0]	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors

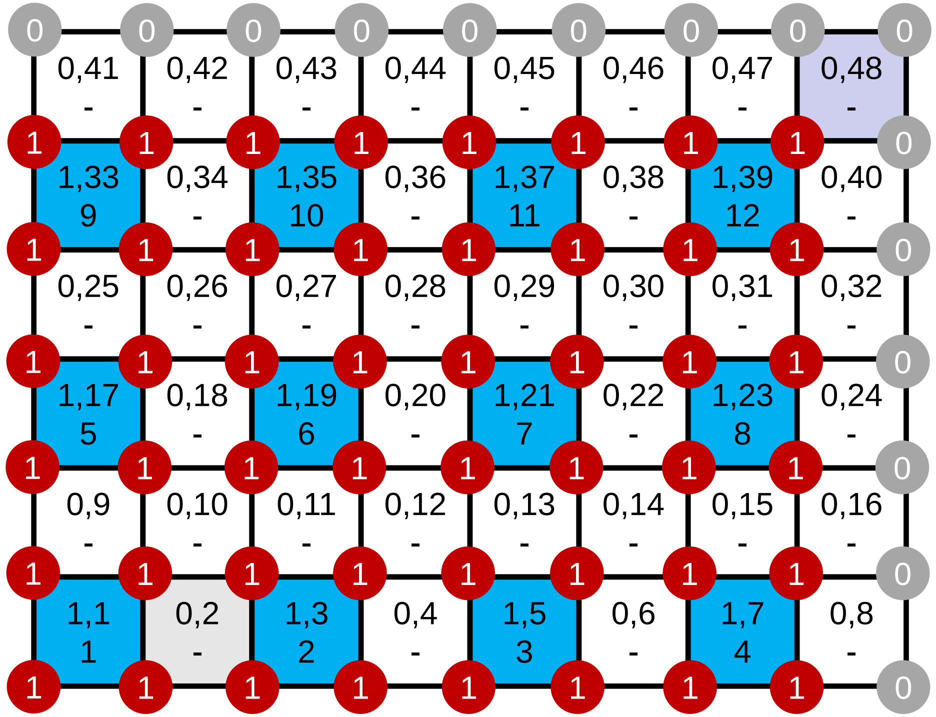


Coloring (2D) (5/7)

```
      allocate (ELMCOLORindex(0:NP))

      allocate (E
      allocate (W

      W1=0; W2=0;
      icou= 0
      do icol= 1, NP
        do i= 1, NP
          W1(i)= 0
        enddo
        do icel= 1, ICELTOT
          if (W2(icel).eq.0) then
            in1= ICELNOD(icel,1)
            in2= ICELNOD(icel,2)
            in3= ICELNOD(icel,3)
            in4= ICELNOD(icel,4)

            ip1= W1(in1)  (=1)
            ip2= W1(in2)  (=0)
            ip3= W1(in3)  (=0)
            ip4= W1(in4)  (=0)

            isum= ip1 + ip2 + ip3 + ip4  (=1)
            if (isum.eq.0) then
              icou= icou + 1
              W3(icol)= icou
              W2(icel)= icol

              ELMCOLORitem(icou)= icel
              W1(in1)= 1
              W1(in2)= 1
              W1(in3)= 1
              W1(in4)= 1
            endif
            if (icou.eq.ICELTOT) goto 100
          endif
        enddo
      enddo
 100
```

icol=1
icel=ICELTOT (=48)

Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 0 1	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 0	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors

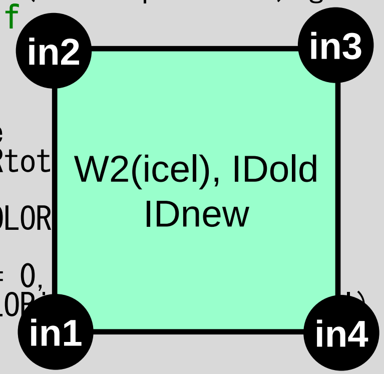


- ✓ Elements in a same color do not share a node
- ✓ Parallel operations are possible for elements in each color

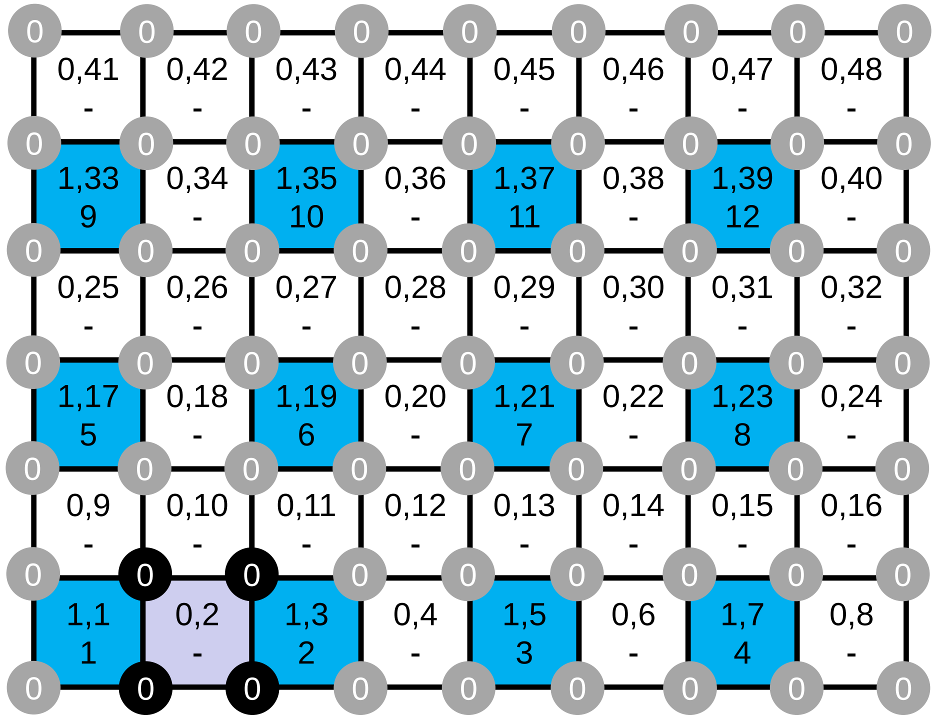
Coloring (2D) (6/7)

```
      allocate (ELMCOLORindex(0:NP))

      allocate (ELMCOLORitem (ICELTOT))
      allocate (W1(NP), W2(ICELTOT), W3(NP))

      W1=0; W2=0; W3=0
      icou= 0
      do icol= 1, NP
        do i= 1, NP
          W1(i)= 0
        enddo
        do icel= 1, ICELTOT
          if (W2(icel).eq.0) then
            in1= ICELNOD(icel,1)
            in2= ICELNOD(icel,2)
            in3= ICELNOD(icel,3)
            in4= ICELNOD(icel,4)

            ip1= W1(in1)  (=0)
            ip2= W1(in2)  (=0)
            ip3= W1(in3)  (=0)
            ip4= W1(in4)  (=0)

            isum= ip1 + ip2 + ip3 + ip4
            if (isum.eq.0) then
              icou= icou + 1
              W3(icol)= icou
              W2(icel)= icol

              ELMCOLORitem(icou)= icel
              W1(in1)= 1
              W1(in2)= 1
              W1(in3)= 1
              W1(in4)= 1
            endif
            if (icou.eq.ICELTOT) goto 100
          endif
        enddo
      enddo

 100  continue
      ELMCOLORtot
      W3(0
      W3(ELMCOLOR

      do icol= 0,
        ELMCOLOR
      enddo
```

icol=2
icel=2



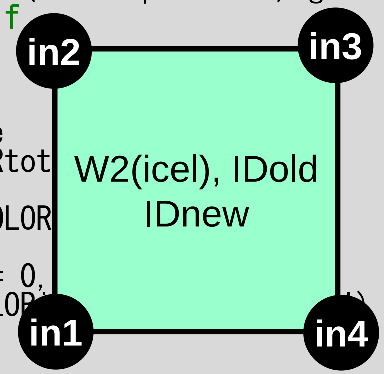
Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 0 1	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 0	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors



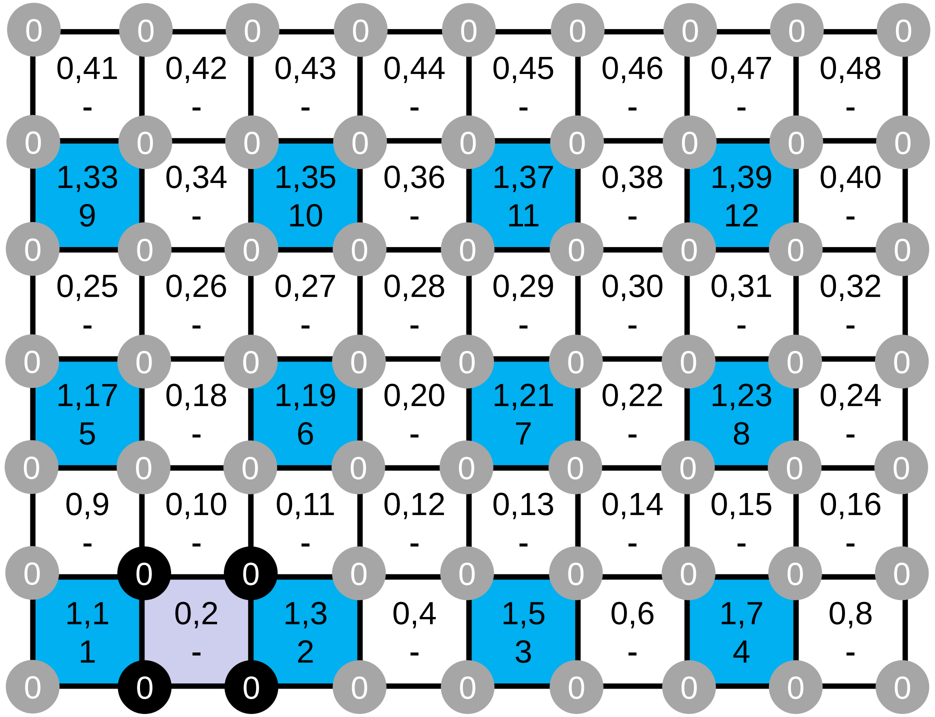
Coloring (2D) (6/7)

```
      allocate (ELMCOLORindex(0:NP))

      allocate (ELMCOLORitem (ICELTOT))
      allocate (W1(NP), W2(ICELTOT), W3(NP))

      W1=0; W2=0; W3=0
      icou= 0
      do icol= 1, NP
        do i= 1, NP
          W1(i)= 0
        enddo
        do icel= 1, ICELTOT
          if (W2(icel).eq.0) then
            in1= ICELNOD(icel,1)
            in2= ICELNOD(icel,2)
            in3= ICELNOD(icel,3)
            in4= ICELNOD(icel,4)

            ip1= W1(in1)  (=0)
            ip2= W1(in2)  (=0)
            ip3= W1(in3)  (=0)
            ip4= W1(in4)  (=0)

            isum= ip1 + ip2 + ip3 + ip4  (=0)
            if (isum.eq.0) then
              icou= icou + 1
              W3(icol)= icou
              W2(icel)= icol

              ELMCOLORitem(icou)= icel
              W1(in1)= 1
              W1(in2)= 1
              W1(in3)= 1
              W1(in4)= 1
            endif
            if (icou.eq.ICELTOT) goto 100
          endif
        enddo
      enddo

 100  continue
      ELMCOLORtot
      W3(0
      W3(ELMCOLOR

      do icol= 0,
        ELMCOLOR
      enddo
```

icol=2
icel=2



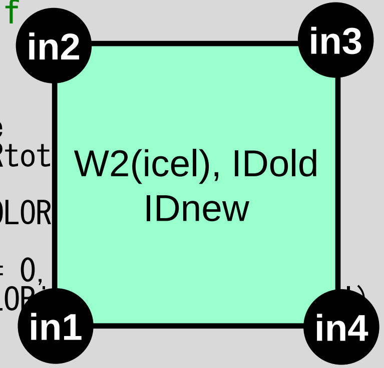
Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 (0) (1)	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 [0]	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors



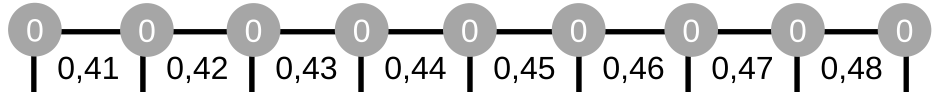
Coloring (2D) (6/7)

```
      allocate (ELMCOLORindex(0:NP))

      allocate (ELMCOLORitem (ICELTOT))
      allocate (W1(NP), W2(ICELTOT), W3(NP))

      W1=0; W2=0; W3=0
      icou= 0
      do icol= 1, NP
        do i= 1, NP
          W1(i)= 0
        enddo
        do icel= 1, ICELTOT
          if (W2(icel).eq.0) then
            in1= ICELNOD(icel,1)
            in2= ICELNOD(icel,2)
            in3= ICELNOD(icel,3)
            in4= ICELNOD(icel,4)

            ip1= W1(in1)  (=0)
            ip2= W1(in2)  (=0)
            ip3= W1(in3)  (=0)
            ip4= W1(in4)  (=0)

            isum= ip1 + ip2 + ip3 + ip4  (=0)
            if (isum.eq.0) then
              icou= icou + 1
              W3(icol)= icou
              W2(icel)= icol

              ELMCOLORitem(icou)= icel
              W1(in1)= 1
              W1(in2)= 1
              W1(in3)= 1
              W1(in4)= 1
            endif
            if (icou.eq.ICELTOT) goto 1
          endif
        enddo
      enddo

 100  continue
      ELMCOLORtot
      W3(0
      W3(ELMCOLOR

      do icol= 0,
        ELMCOLOR
      enddo
```

icol=2
icel=2

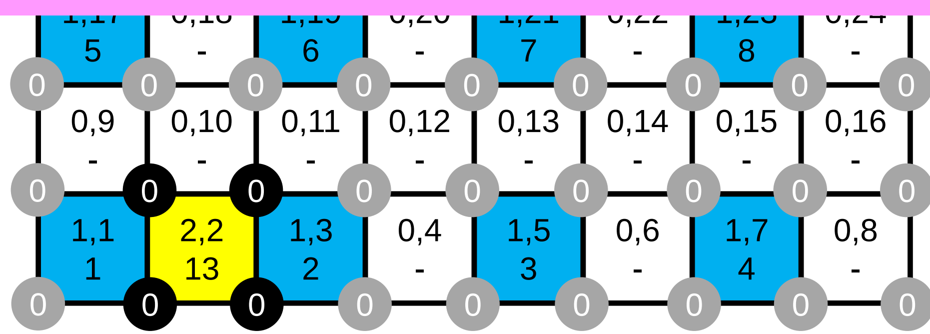


Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 (0) (1)	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 [0]	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors



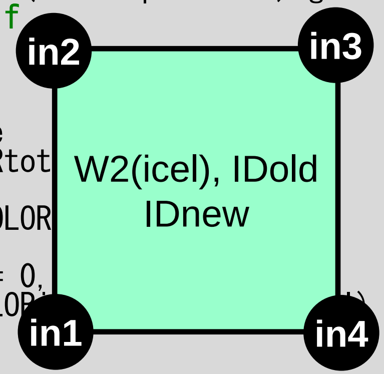
Because no vertices on this element were "flagged" yet in this Color (=icol), this element can join this Color (=icol) !!

icou= icou + 1 Colored Element #, NEW Element ID
W3(icol)= icou Accumulated # of Colored Elem's in Each Color
W2(icel)= icol Color ID of Each Element
ELMCOLORitem(icou)= icel OLD Element ID



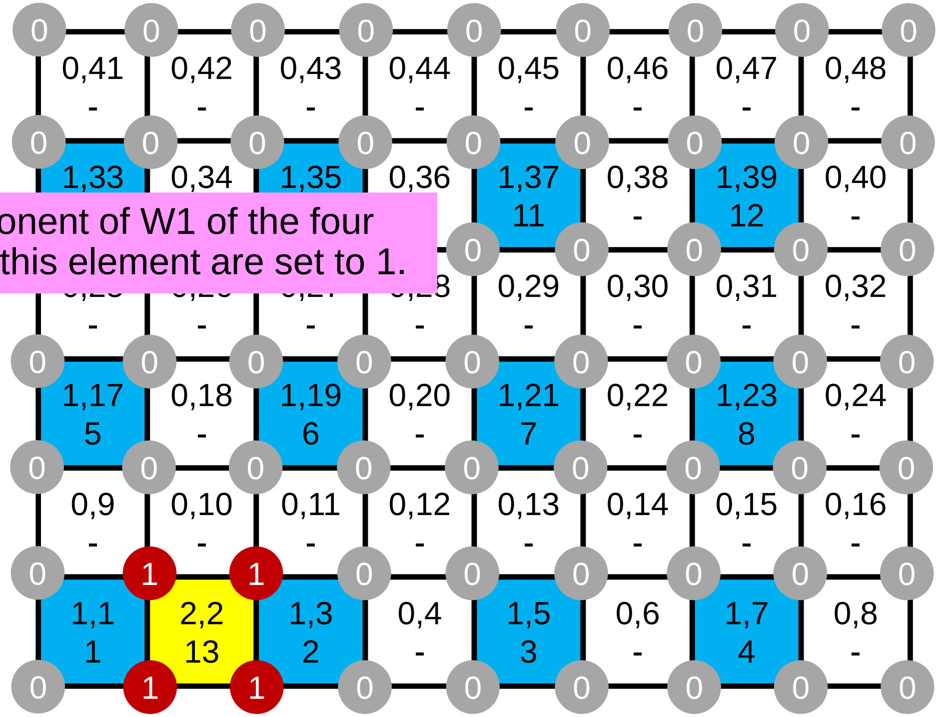
Coloring (2D) (6/7)

```
      allocate (ELMCOLORindex(0:NP))

      allocate (ELMCOLORitem (ICELTOT))
      allocate (W1(NP), W2(ICELTOT), W3(NP))

      W1=0; W2=0; W3=0
      icou= 0
      do icol= 1, NP
        do i= 1, NP
          W1(i)= 0
        enddo
        do icel= 1, ICELTOT
          if (W2(icel).eq.0) then
            in1= ICELNOD(icel,1)
            in2= ICELNOD(icel,2)
            in3= ICELNOD(icel,3)
            in4= ICELNOD(icel,4)

            ip1= W1(in1)  (=0)
            ip2= W1(in2)  (=0)
            ip3= W1(in3)  (=0)
            ip4= W1(in4)  (=0)

            isum= ip1 + ip2 + ip3 + ip4  (=0)
            if (isum.eq.0) then
              icou= icou + 1
              W3(icol)= icou
              W2(icel)= icol

              ELMCOLORitem(icou)= icel
              W1(in1)= 1
              W1(in2)= 1
              W1(in3)= 1
              W1(in4)= 1
            endif
            if (icou.eq.ICELTOT) goto 100
          endif
        enddo
      enddo

 100  continue
      ELMCOLORtot
      W3(0
      W3(ELMCOLOR

      do icol= 0,
        ELMCOLOR
      enddo
```



icol=2
icel=2

Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 0 1	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 0	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors

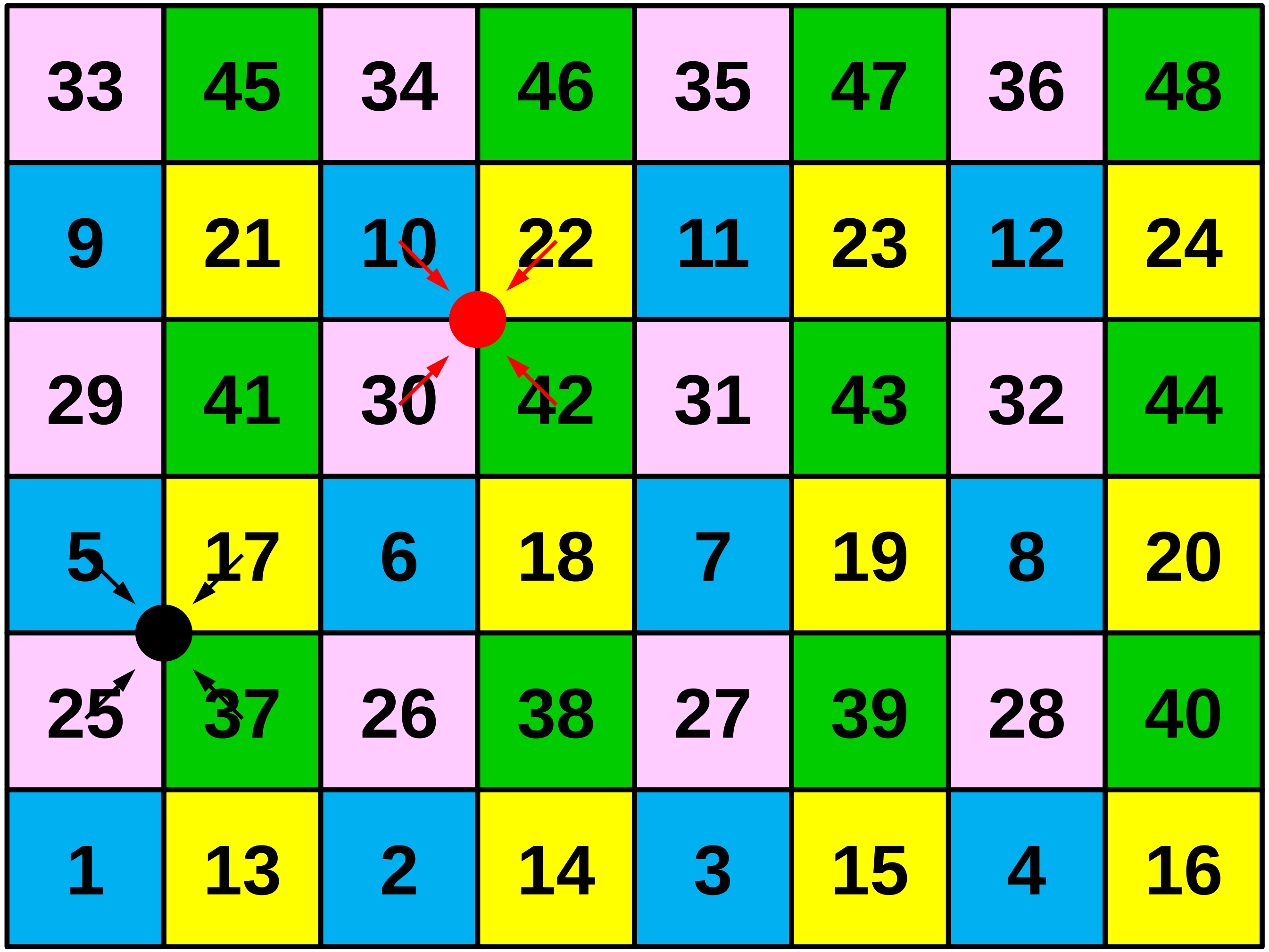


Each component of W1 of the four vertices on this element are set to 1.

Multi-Threading: Mat_Ass

Parallel operations are possible for elements in same color (they are independent)

Colors of elements sharing a node are different



Coloring (2D) (7/7)

```
      allocate (ELMCOLORindex(0:NP))

      allocate (ELMCOLORitem (ICELTOT))
      allocate (W1(NP), W2(ICELTOT), W3(NP))

      W1=0; W2=0; W3=0
      icou= 0
      do icol= 1, NP
        do i= 1, NP
          W1(i)= 0
        enddo
        do icel= 1, ICELTOT
          if (W2(icel).eq.0) then
            in1= ICELNOD(icel,1)
            in2= ICELNOD(icel,2)
            in3= ICELNOD(icel,3)
            in4= ICELNOD(icel,4)

            ip1= W1(in1)
            ip2= W1(in2)
            ip3= W1(in3)
            ip4= W1(in4)

            isum= ip1 + ip2 + ip3 + ip4
            if (isum.eq.0) then
              icou= icou + 1
              W3(icol)= icou
              W2(icel)= icol

              ELMCOLORitem(icou)= icel
              W1(in1)= 1
              W1(in2)= 1
              W1(in3)= 1
              W1(in4)= 1
            endif
            if (icou.eq.ICELTOT) goto 100
          endif
        enddo
      enddo

 100  continue
      ELMCOLORtot= icol
      W3(0          )= 0
      W3(ELMCOLORtot)= ICELTOT

      do icol= 0, ELMCOLORtot
        ELMCOLORindex(icol)= W3(icol)
      enddo
```

Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1 0 1	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2 0	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem's in each Color
ELMCOLORtot		Total # of Colors

33	45	34	46	35	47	36	48
9	21	10	22	11	23	12	24
29	41	30	42	31	43	32	44
5	17	6	18	7	19	8	20
25	37	26	38	27	39	28	40
1	13	2	14	3	15	4	16

Multi-Threaded Matrix Assembling Procedure

```
      do icol= 1, ELMCOLORtot
!$omp parallel do private (icel0,icel)                                  &
!$omp&             private (in1,in2,in3,in4,in5,in6,in7,in8)           &
!$omp&             private (nodLOCAL,ie,je,ip,jp,kk,iiS,iiE,k)          &
!$omp&             private (DETJ,PNX,PNY,PNZ,QVC,QV0,COEFij,coef,SHi)   &
!$omp&             private (PNXi,PNYi,PNZi,PNXj,PNYj,PNZj,ipn,jpn,kpn)  &
!$omp&             private (X1,X2,X3,X4,X5,X6,X7,X8)                    &
!$omp&             private (Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8)                    &
!$omp&             private (Z1,Z2,Z3,Z4,Z5,Z6,Z7,Z8,COND0)
      do icel0= ELMCOLORindex(icol-1)+1, ELMCOLORindex(icol)
        icel= ELMCOLORitem(icel0)
        in1= ICELNOD(icel,1)
        in2= ICELNOD(icel,2)
        in3= ICELNOD(icel,3)
        in4= ICELNOD(icel,4)
        in5= ICELNOD(icel,5)
        in6= ICELNOD(icel,6)
        in7= ICELNOD(icel,7)
        in8= ICELNOD(icel,8)
...
```

icel0: NEW Elem. ID, **icel**: OLD Elem. ID

Name	Size	Content
ELMCOLORindex	**0:NP**	Number of Elements in Each Color
ELMCOLORitem	**ICELTOT**	OLD Element ID in Each Color
W1	**NP**	Flag of Each Node =0: Not flagged in the current color =1: Already Flagged
W2	**ICELTOT**	Color ID of Each Element
W3	**0:NP**	Accumulated # of Colored Elem’s in each Color
ELMCOLORtot		Total # of Colors

How to apply multi-threading

- CG Solver
  - OpenMP指示文を挿入するのみ
  - ILU/IC 前処理の場合はもっと難しい
- MAT_ASS (mat_ass_main, mat_ass_bc)
  - データ依存性あり
  - 複数要素による1節点への足し込みが並列計算時に同時に発生することを避ける必要がある
    - 答えが変わる，もしくはDead Lockが生じる可能性がある
  - 色づけ：Coloring
    - 同じ色に彩色された要素は節点を共有しない
    - 同じ色の要素には並列計算が可能
    - 本問題の場合，三次元では8色，二次元では4色必要
    - 色づけ部分の計算はexpensive：並列化困難