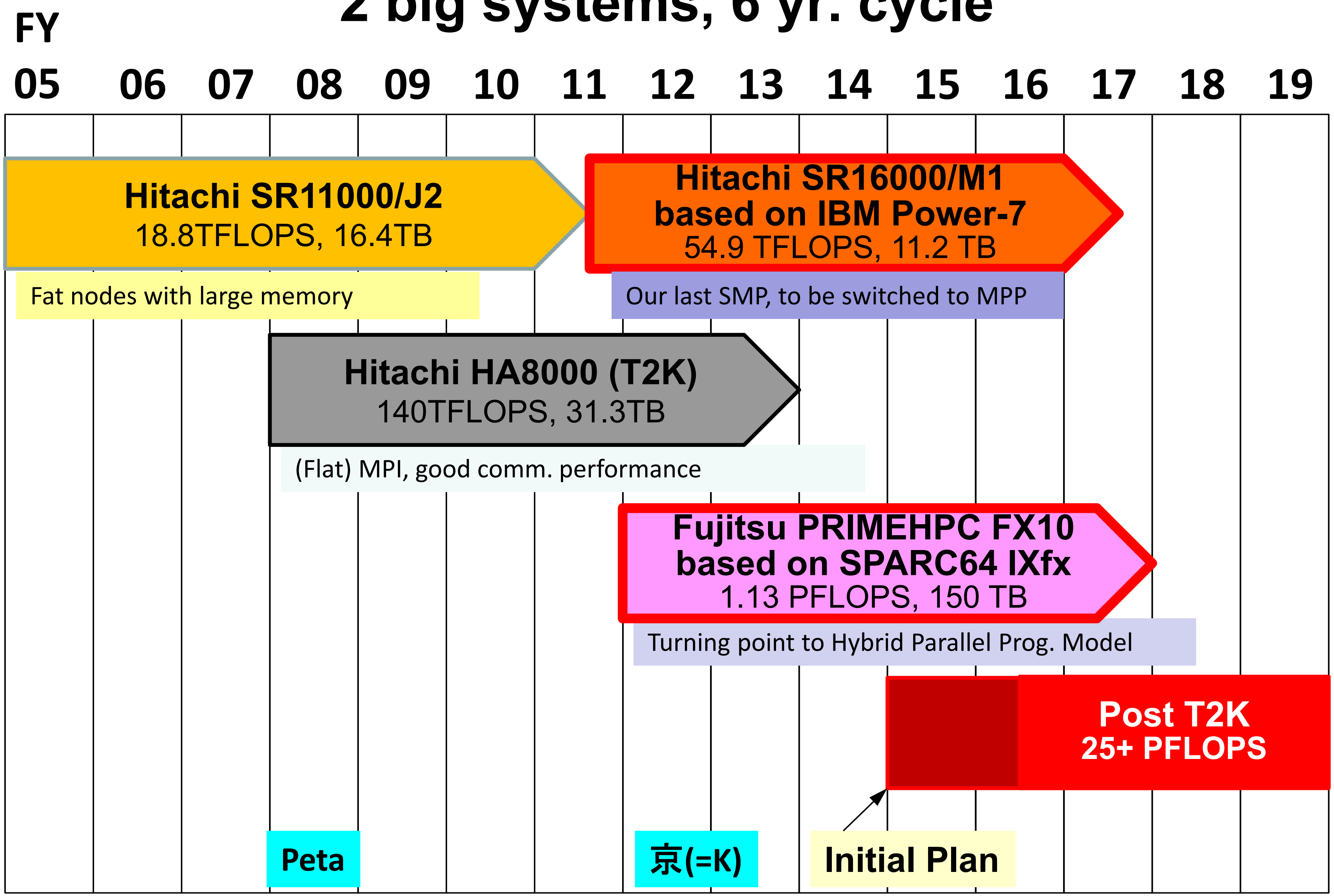
Experiences in Optimizations of Preconditioned Iterative Solvers for FEM/FVM Applications & Matrix Assembly of FEM using Intel Xeon Phi

Kengo Nakajima
Supercomputing Research Division
Information Technology Center, The University of Tokyo

IXPUG BoF at SC15, November 18, 2015, Austin, Texas

Supercomputers in U.Tokyo

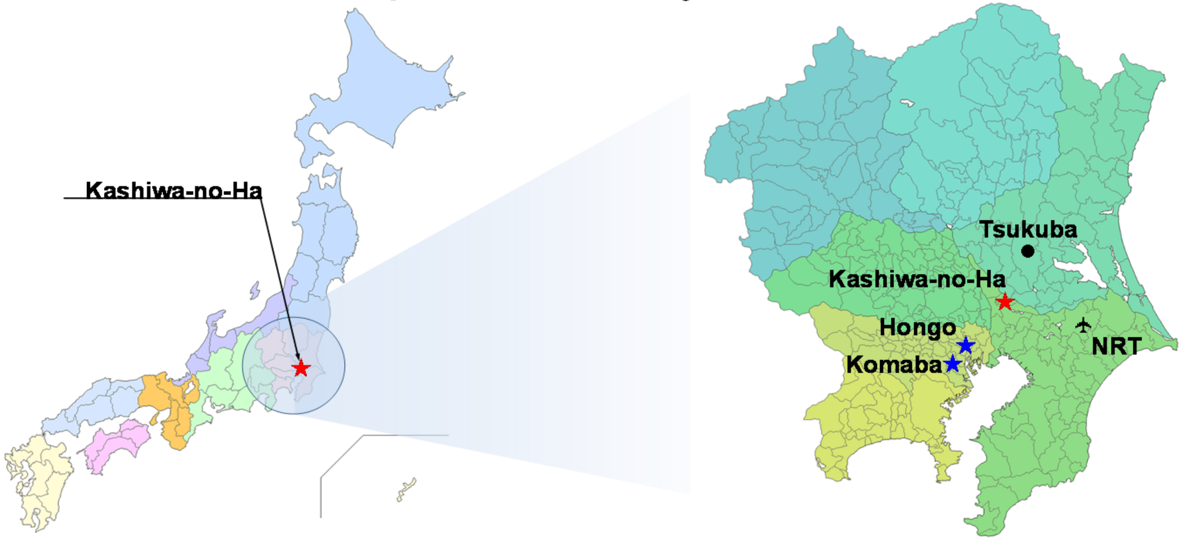
2 big systems, 6 yr. cycle



Post T2K System

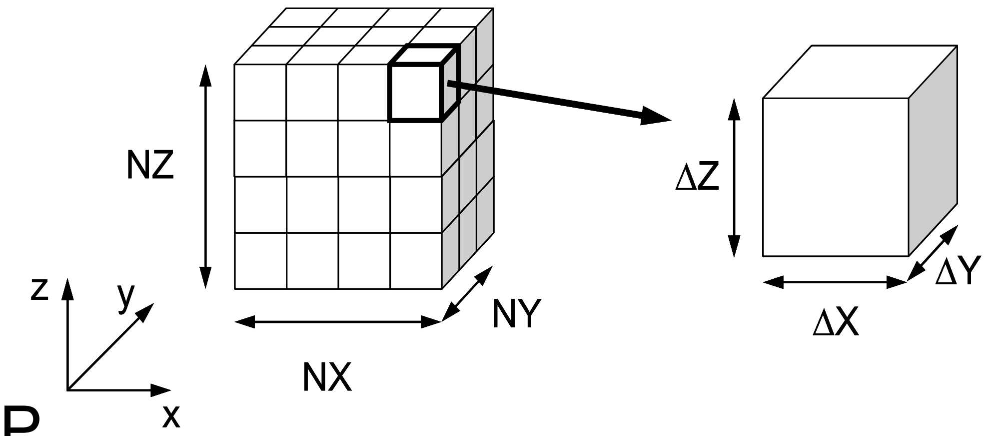
- 25+ PFLOPS, FY. 2016
- Many-core based (e.g. KNL)
- Joint Center for Advanced High Performance Computing (JCAHPC, http://jcahpc.jp/)
  - University of Tsukuba
  - University of Tokyo
    - New system will installed in Kashiwa-no-Ha (Leaf of Oak) Campus/U.Tokyo, which is between Tokyo and Tsukuba

JCAHPC



Target Applications
Sparse Matrices = Memory Bound

- GeoFEM Cube
  - 3D FEM
  - Solid Mechanics & Heat Transfer
  - Preconditioned Iterative Solver (PCG, Point Jacobi)
  - Performance
    - PCG Solver & Matrix Assembly
- Poisson3D-OMP
  - 3D FVM
  - Steady Poisson Eq.
  - Performance
    - ICCG Solver
- Codes
  - Simple Geometries
  - Fortran 90, MPI + OpenMP



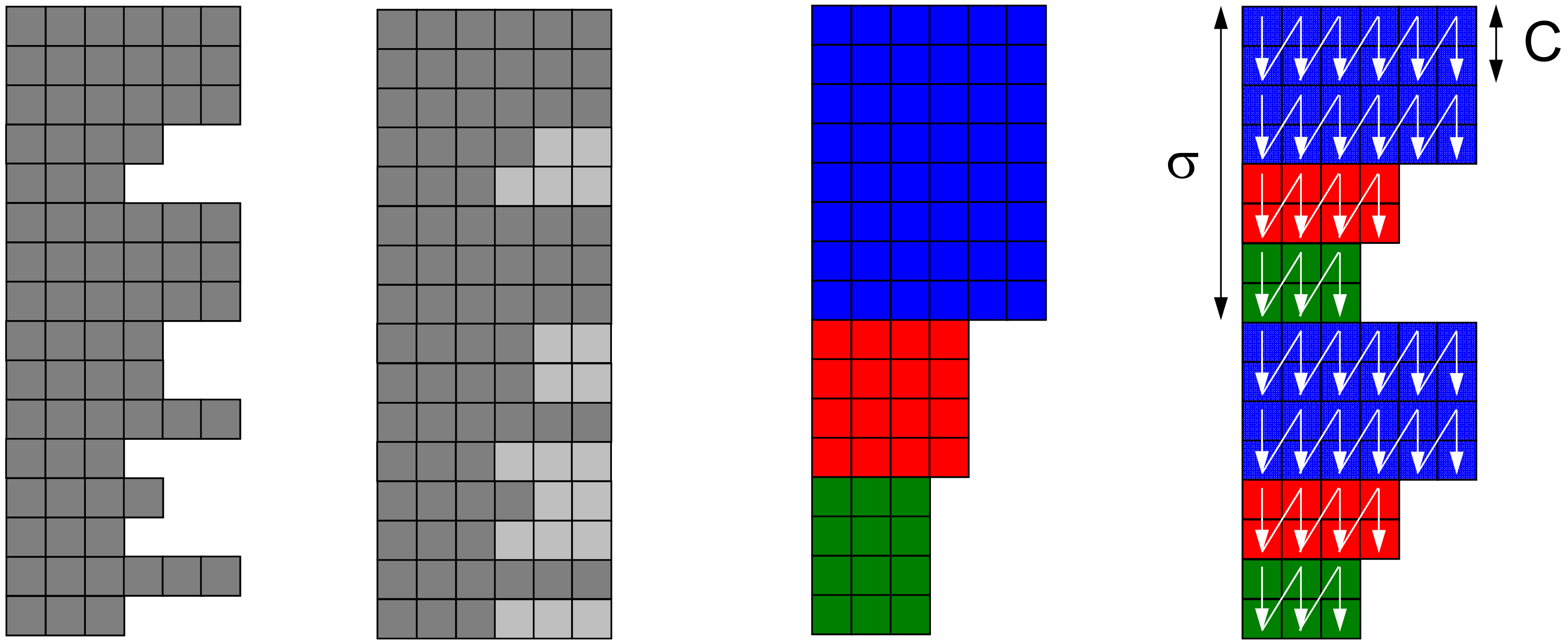
H/W: Single Node/Socket

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
STREAM Triad (GB/s)	64	159	49
Cache	L1:32KB/core L2:12MB/socket	L1:32KB/core L2:512KB/core	L1:32KB/core L2:256KB/core L3:25MB/socket

What to be evaluated ?

- Effects of formats for storing sparse matrices in PCG/ICCG Solvers

[Kreutzer, Hager, Wellein 2014]



CRS | ELL | Sliced ELL | SELL-C-σ (SELL-2-8)

- Effects of Vectorization, !$omp simd

- **PCG: GeoFEM Cube/Heat**
- ICCG: Poisson3D-OMP
- Matrix Assembly: GeoFEM Cube/Solid

Example: SELL-4-σ GeoFEM Cube/Heat: SpMV

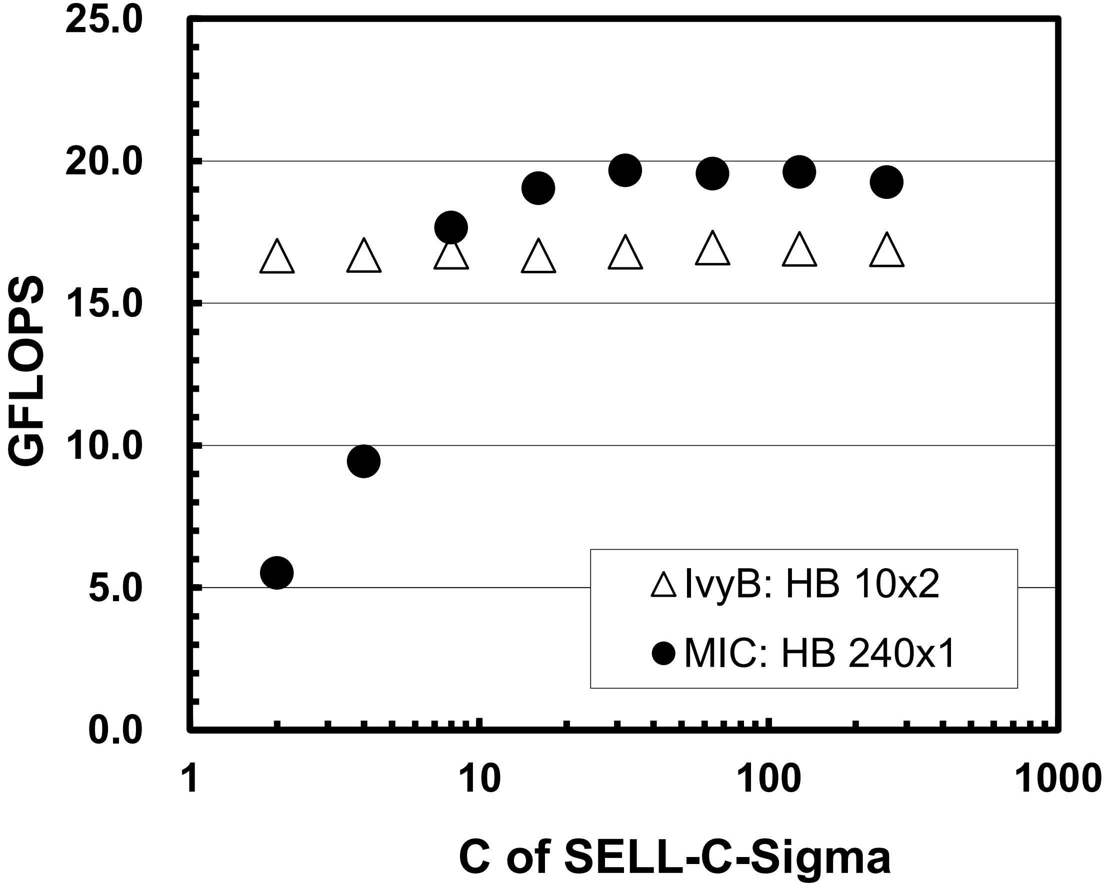
```
!$omp parallel do
      do i= 1, N/2
        Y(2*(i-1)+1)= D(2*(i-1)+1) * X(2*(i-1)+1)
        Y(2*(i-1)+2)= D(2*(i-1)+2) * X(2*(i-1)+2)
      enddo

!$omp parallel do
      do i= 1, N/4
      do j= 1, NCOL(i)
!$omp simd
      do k= 1, 4
        Y(4*(i-1)+k)= Y(4*(i-1)+k) + AMAT(CS(i-1)+(j-1)*4+k)      &
                                   * X(COL(CS(i-1)+(j-1)*4+k))
      enddo
      enddo
      enddo
```

Width of Chunk: l_i (Number of Non-Zero Off Diagonals: **`CL(i)`**)

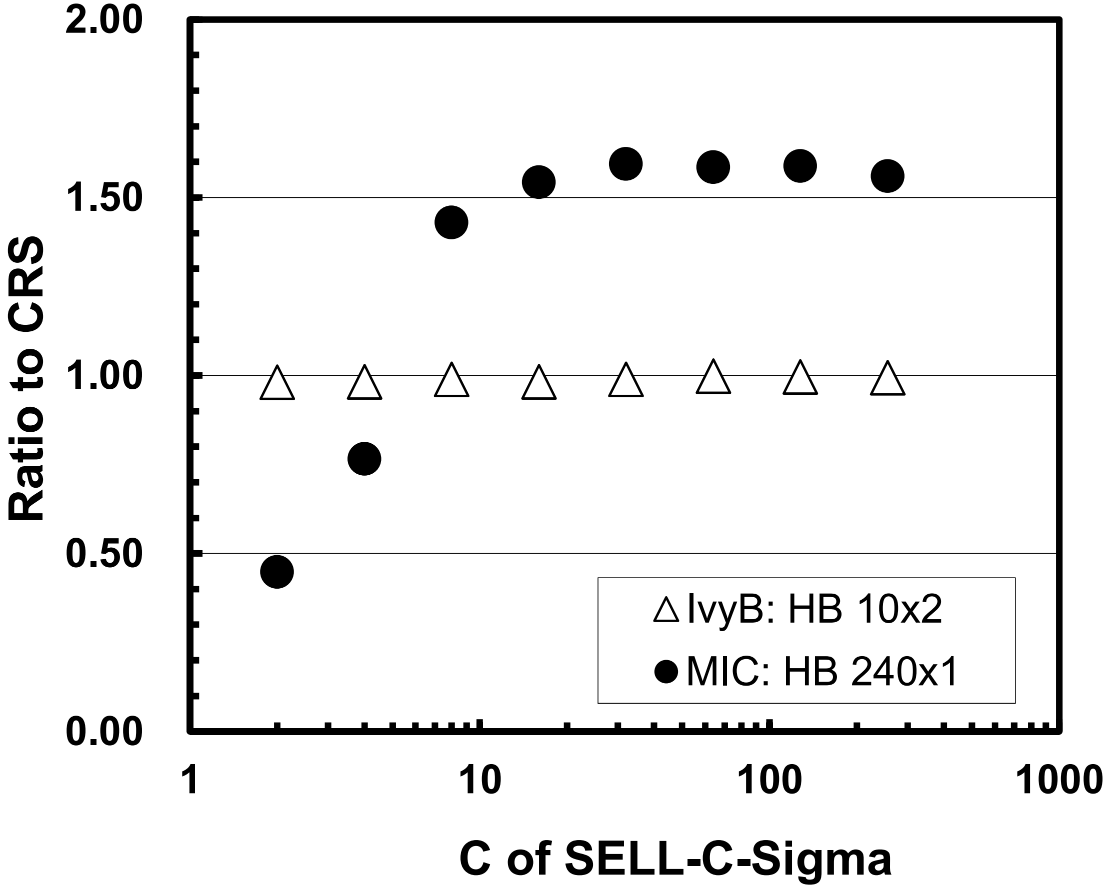
PCG in GeoFEM Cube/Heat GFLOPS Rate

MIC：Low Performance if C（Chunk Size）< 8



- IvyB
  - 10 threads x 2 soc.
- MIC
  - 240 threads x 1
  - SIMD幅512bit
  - 64bit×8

PCG in GeoFEM Cube/Heat Ratio to CRS (Original)

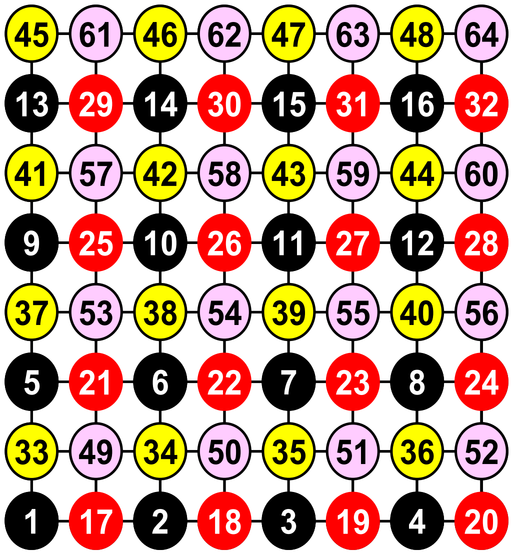


- IvyB
  - No effects by SELL-C-Sigma/ELL
  - Out-of-Order ?

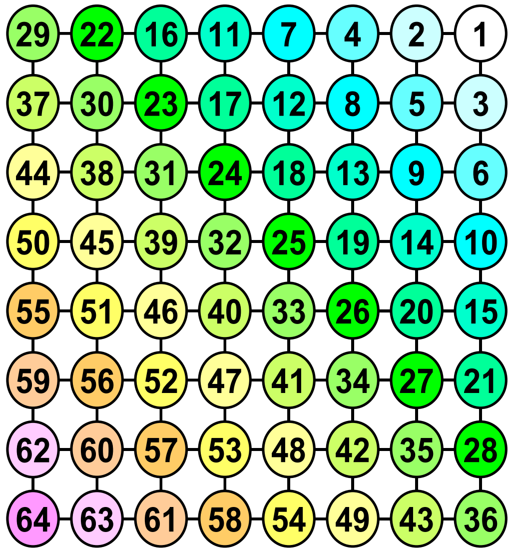
- PCG: GeoFEM Cube/Heat
- **ICCG: Poisson3D-OMP**
- Matrix Assembly: GeoFEM Cube/Solid

Elimination of Data Dependency in ICCG

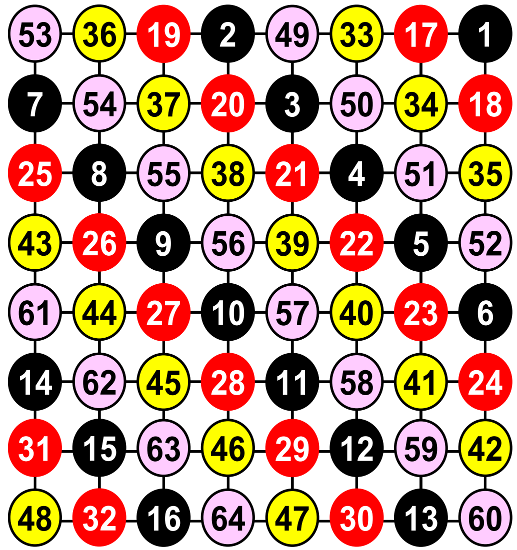
- Coloring + Reordering
- More colors, better convergence/larger sync. overhead
  - MC: good parallel performance, bad convergence
  - RCM: good convergence, many colors
  - **CM-RCM: combinations of MC + RCM**



MC (Color#=4)
Multicoloring



RCM
Reverse Cuthill-Mckee

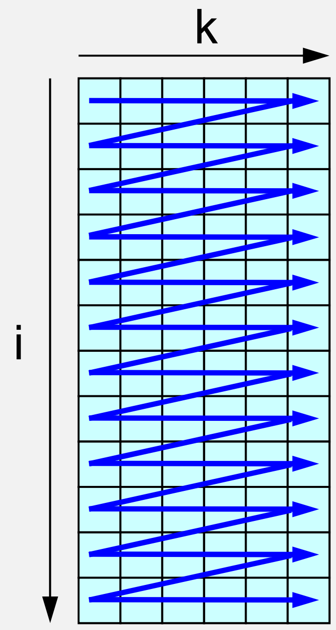


CM-RCM (Color#=4)
Cyclic MC + RCM

ELL: Row-wise

CRS with fixed length
Forward Substitution

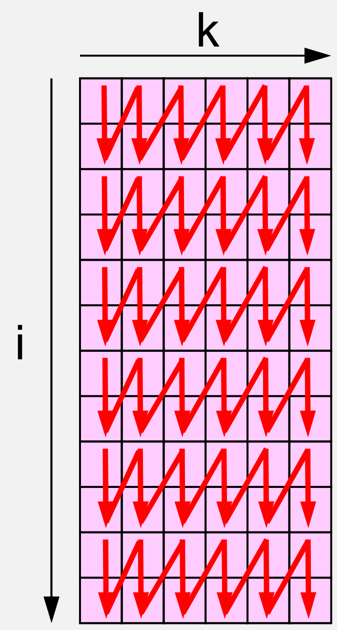
```
!$omp parallel
    do icol= 1, NCOLORtot
!$omp do
    do ip  = 1, PEsmpTOT
      do i= Index(ip-1,icol)+1, Index(ip,icol)
        do k= 1, 6
          Z(i)= Z(i) - AML(k,i)*Z(IAML(k,i))
        enddo
        Z(i)= Z(i) / DD(i)
      enddo
    enddo
    enddo
!omp end parallel
```



ELL: Column-wise

Jagged Diagonal
Forward Substitution

```
!$omp parallel
    do icol= 1, NCOLORtot
!$omp do
    do ip  = 1, PEsmpTOT
      do k= 1, 6
!$omp simd
        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
```



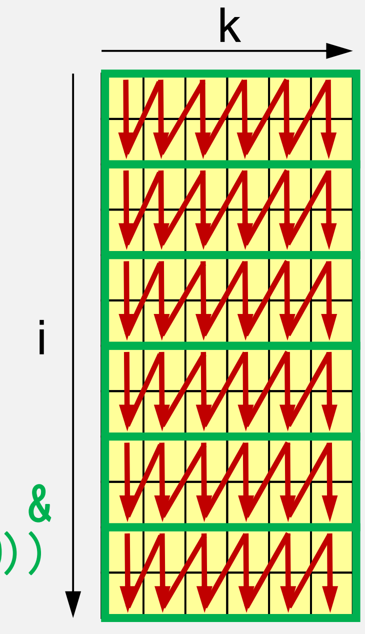
Each Color/Thread

ELL: Column-wise

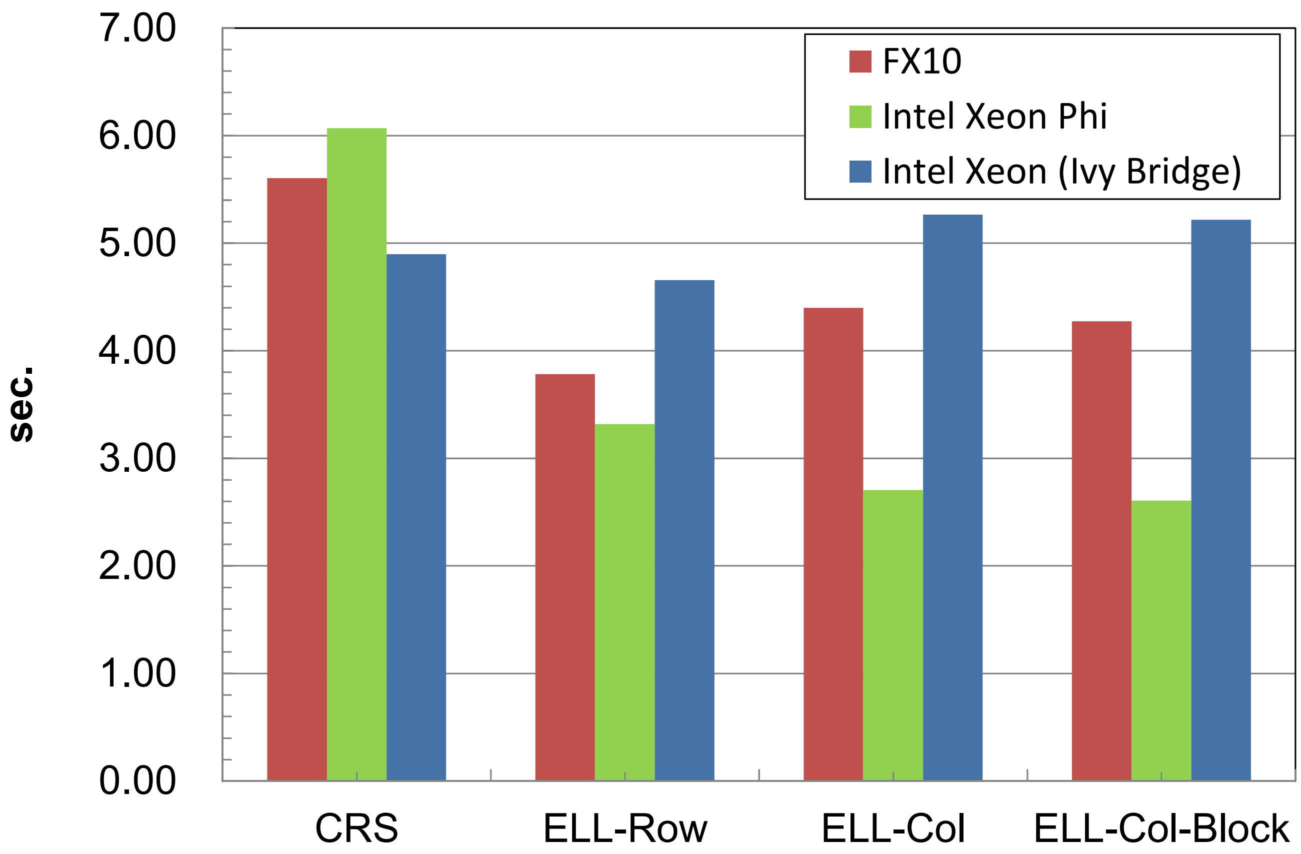
Jagged Diagonal, Blocked Version

Forward Substitution

```
!$omp parallel
    do icol= 1, NCOLORtot
!$omp do
    do ip  = 1, PEsmpTOT
      blkID= (ip-1)*NCOLORtot + ip
      do k= 1, 6
!$omp simd
      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
```

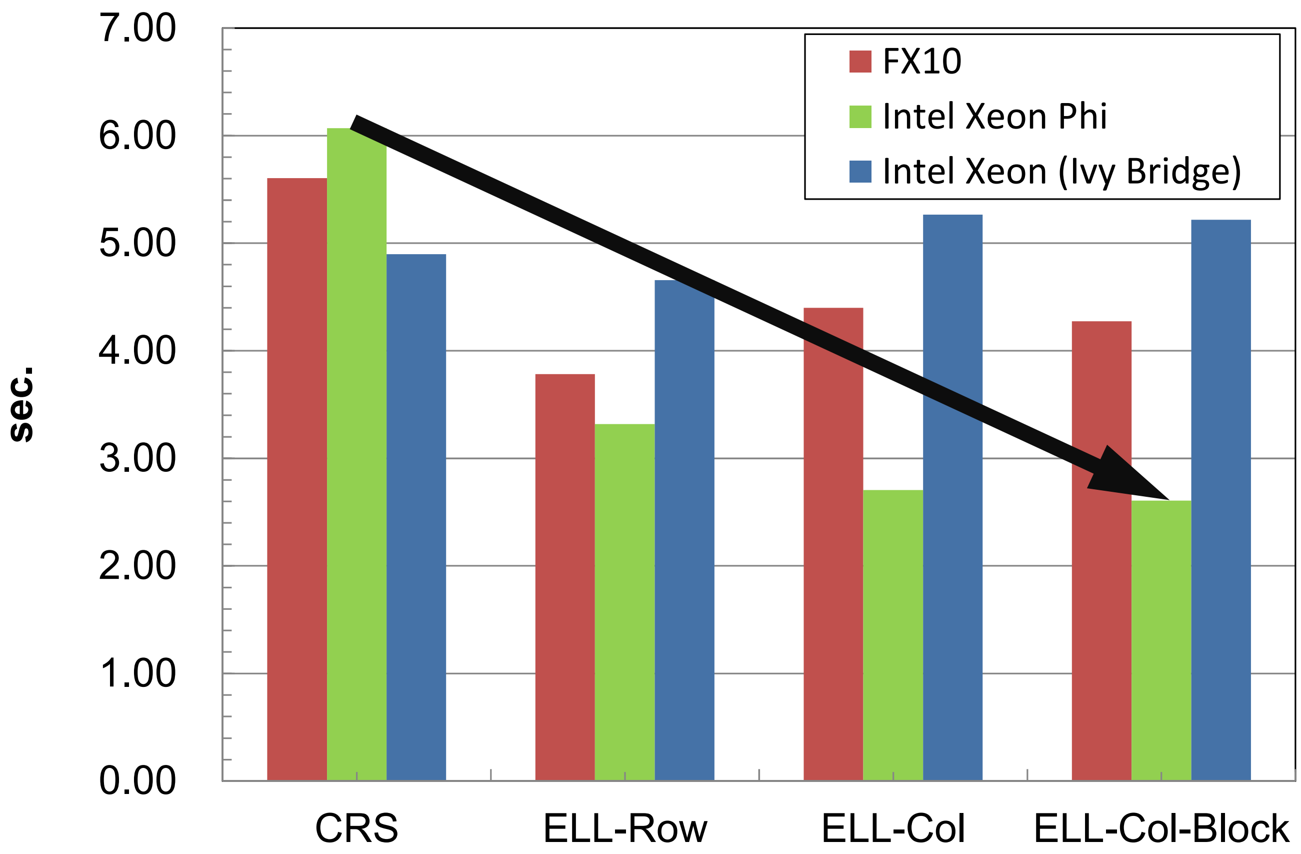


Results (Computation Time for Linear Solver): Down is Good !



- FX10
  - 16 T x 1
- IvyB
  - 10 T x 1
- MIC
  - 240 T x 1

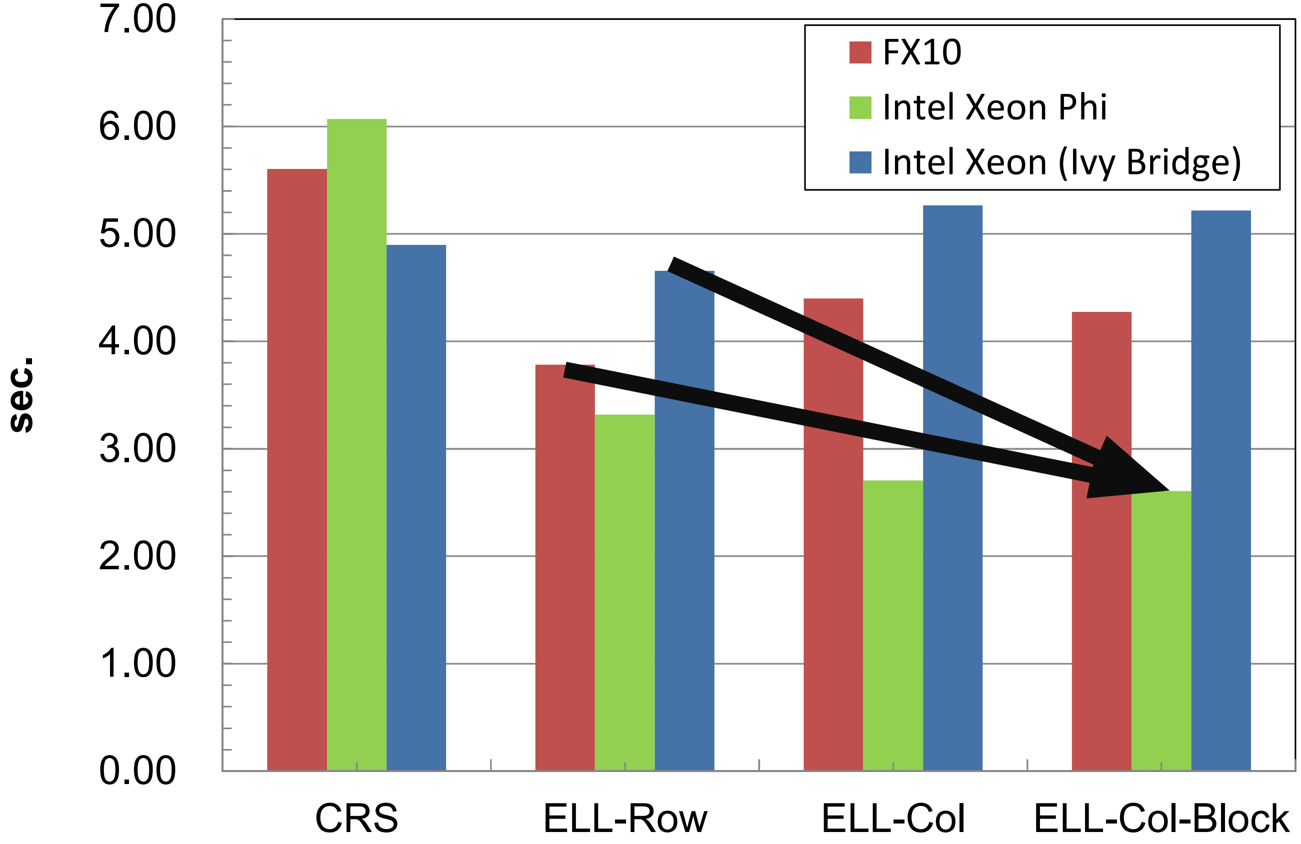
Results (Computation Time for Linear Solver): Down is Good !



- FX10
  - 16 T x 1
- IvyB
  - 10 T x 1
- MIC
  - 240 T x 1

2.33x Improvement
CRS -> ELL-Col-Block
Contributions by !$omp simd: 5-10%

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



- FX10
  - 16 T x 1
- IvyB
  - 10 T x 1
- MIC
  - 240 T x 1

1.45x faster than FX10
1.79x faster than IvyB

Compiler Options for MIC: Additional Improvement

ELL-Col-Block	Coalesced		Sequential	
	sec.	GFLOPS	sec.	GFLOPS
-O3 -openmp -mmic -align array64byte (base)	2.654	10.53	**2.603**	**10.74**
-opt-streaming-stores always	3.165	8.832	3.159	8.849
-opt-streaming-cache-evict=0	2.625	10.65	2.600	10.75
-opt-streaming-cache-evict=1	2.639	10.59	2.605	10.73
-opt-streaming-stores always -opt-streaming-cache-evict=0	2.486	11.24	2.539	11.01
-opt-streaming-stores always -opt-streaming-cache-evict=1	2.477	11.29	2.556	10.94
-opt-streaming-stores always -opt-streaming-cache-evict=0 -opt-prefetch-distance=a,b	**2.385 (2,0)**	**11.72**	2.477 (8,1)	11.29
-opt-streaming-stores always -opt-streaming-cache-evict=1 -opt-prefetch-distance=c,d	2.404 (2,0)	11.63	2.487 (16,1)	11.24

Further Optimization by Compiler Options

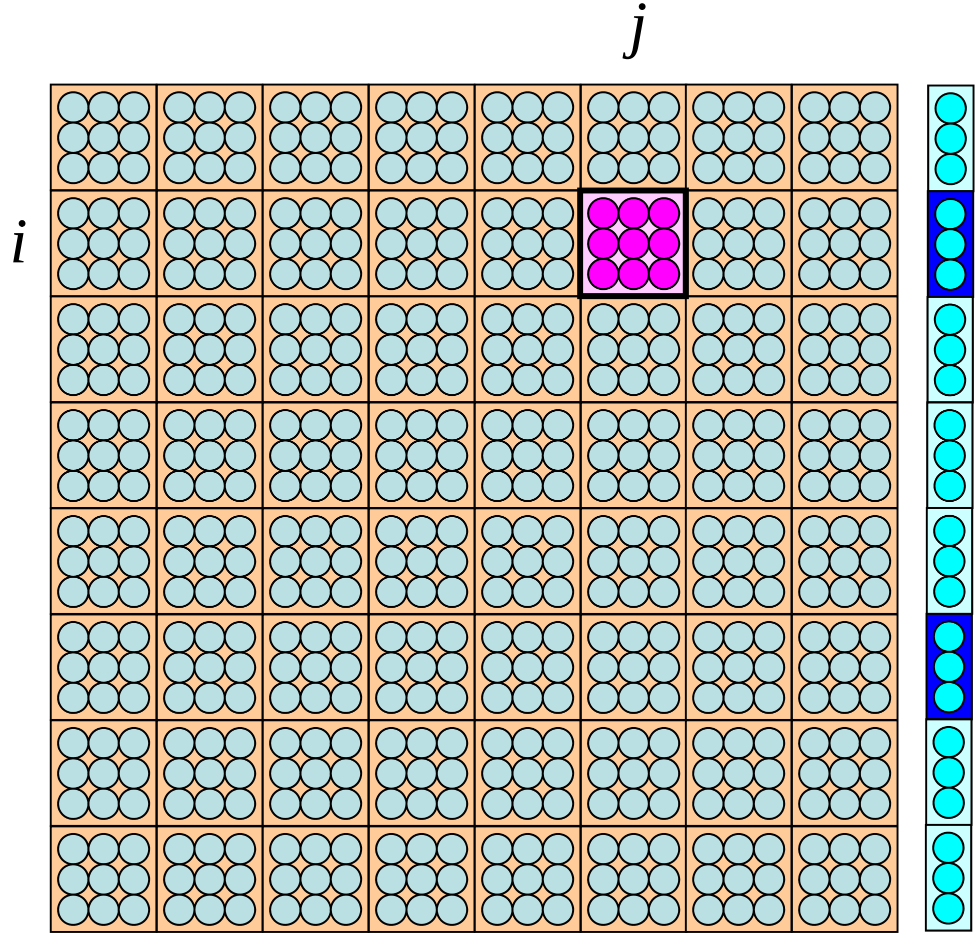
✓-opt-streaming-cache-evict=0	+1.14%
✓-opt-streaming-stores always	+5.60%
✓-opt-prefetch-distance=a,b	+4.56%
	(11.3% total)

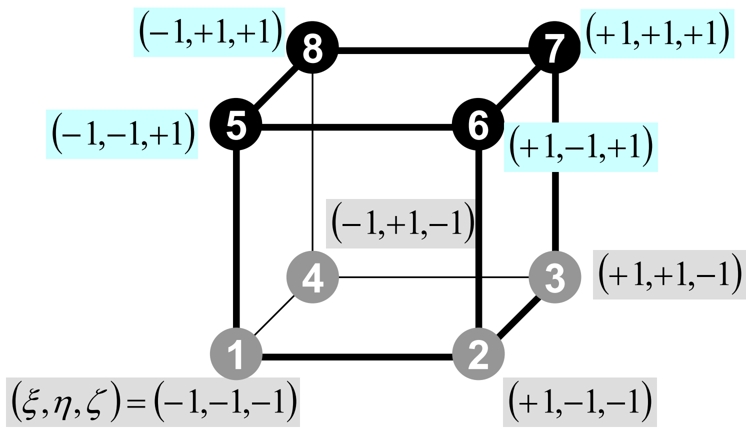
✓Automatic experimental tool for compiler options is now under development

- PCG: GeoFEM Cube/Heat
- ICCG: Poisson3D-OMP
- **Matrix Assembly: GeoFEM Cube/Solid**

Element Matrix: 24x24 Dense Mat.

(u,v,w) components on each node are physically strongly-coupled: these three components are treated in block-wise manner: 8×8 matrix





$$\begin{bmatrix} a_{i_e j_e 11} & a_{i_e j_e 12} & a_{i_e j_e 13} \\ a_{i_e j_e 21} & a_{i_e j_e 22} & a_{i_e j_e 23} \\ a_{i_e j_e 31} & a_{i_e j_e 32} & a_{i_e j_e 33} \end{bmatrix} \left(i_e, j_e = 1 \ldots 8\right)$$

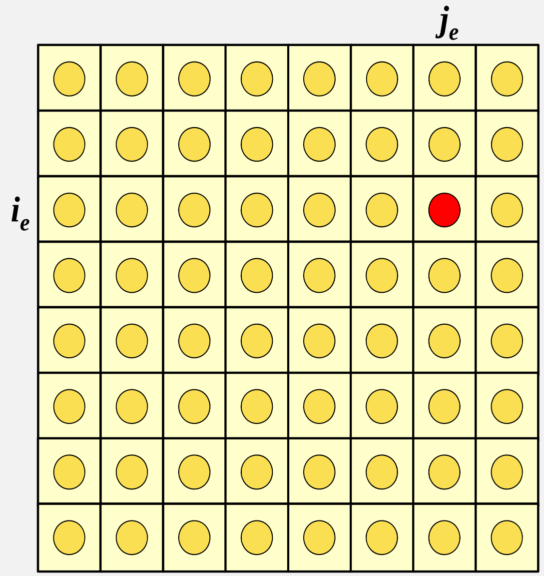
Overview of Matrix Assembling

```
do icel= 1, ICELTOT      Loop for Elements

  <Calculates Jacobian>

  do ie= 1, 8          Local ID
    do je= 1, 8        Local ID
      <Global ID of Nodes: ip, jp>
      <Address of A_ip,jp in sparse matrix: kk>

      do kpn= 1, 2      Gaussian Quad. in ζ
        do jpn= 1, 2    Gaussian Quad. in η
          do ipn= 1, 2  Gaussian Quad. in ξ
            <Calculation of Element Matrices>
          enddo
        enddo
      enddo
    enddo
  enddo
  <Accumulation to Global Matrix>
enddo
```



$$\left[k_{i_e j_e}\right] \quad (i_e, j_e = 1 \sim 8)$$

Assembly of Matrices

- ① Calculations of Jacobian and Grad. of Shape Fn's
- ② Searching of Addresses of Sparse Matrices
- ③ Calculations of Element Matrices
- ④ Accumulations to Global Matrix
  - COLORtot: Number of Colors for Elements (=8)
  - col_index(color): Number of Elements in each Color

```
    do color= 1, COLORtot
!$OMP PARALLEL DO
    do icel= col_index(color-1)+1, col_index(color)
      <① Jacobian & Grad. Shape Fn' s>
      do ie= 1, 8; do je= 1, 8
        <② Address in Sparse Matrix >
        <③ Element Matrices>
        <④ Global Matrix>
      enddo; enddo
    enddo
    enddo
```

Type-A

BLKSIZ: Elem.# in Blocks, NBLK: Block #, icel: Elem. ID

```
!$omp parallel (…)
    do color= 1, COLORtot
!$omp do
    do ip= 1, THREAD_num
      NBLK: calculated by (col_index, color, thread#)
    do ib= 1, NBLK
      do blk= 1, BLKSIZ
        icel: calculated by (col_index, ib, blk)
!$omp simd
        do ie= 1, 8; do je= 1, 8
          <② Address in Sparse Matrix>
        enddo; enddo
      enddo
      do blk= 1, BLKSIZ
        icel: calculated by (col_index, ib, blk)
        <① Jacobian & Grad. Shape Fn’ s>
!$omp simd
        do ie= 1, 8; do je= 1, 8
          <③ Element Matrices >
        enddo; enddo
      enddo
      do blk= 1, BLKSIZ
        icel: calculated by (col_index, ib, blk)
!$omp simd
        do ie= 1, 8; do je= 1, 8
          <④ Global Matrices>
        enddo; enddo
      enddo
    enddo
    enddo
    enddo
!$omp end parallel
```

Type-B

BLKSIZ: Elem.# in Blocks, NBLK: Block #, icel: Elem. ID

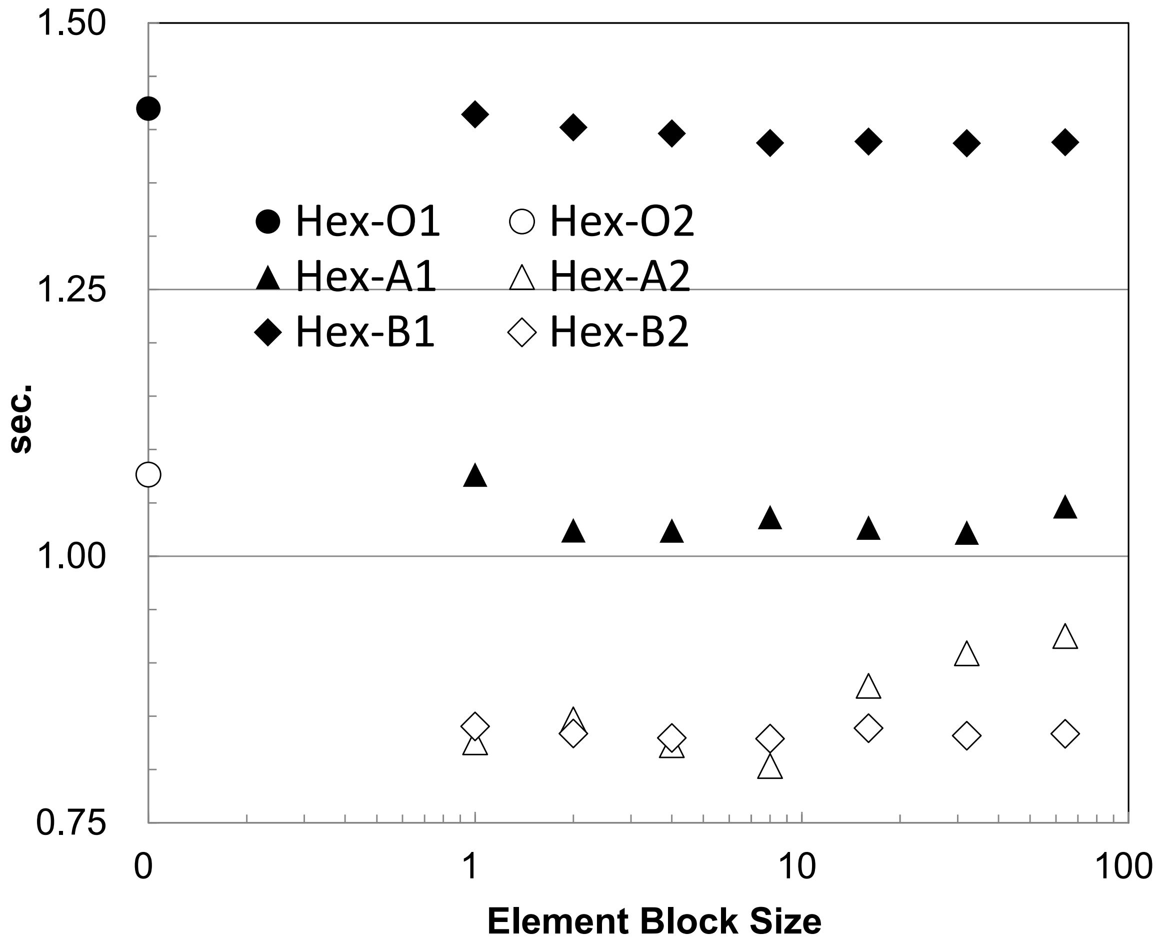
```
!$omp parallel (…)
    do color= 1, COLORtot
!$omp do
    do ip= 1, THREAD_num
      NBLK: calculated by (col_index, color, thread#)
    do ib= 1, NBLK
      do blk= 1, BLKSIZ
        icel: calculated by (col_index, ib, blk)
!$omp simd
        do ie= 1, 8; do je= 1, 8
          <② Address in Sparse Matrix>
        enddo; enddo
      enddo
      do blk= 1, BLKSIZ
        <① Jacobian & Grad. Shape Fn’s>
        icel: calculated by (col_index, ib, blk)
!$omp simd
        do ie= 1, 8; do je= 1, 8
          <③ Element Matrices>
          <④ Global Matrices>
        enddo; enddo
      enddo
    enddo
    enddo
    enddo
!$omp end parallel
```

Comp. Time for Matrix Assembly

KNC: 240 Threads x 1

O: Original, A: Type-A, B: Type-B

1: without !$omp simd, 2: with !$omp simd



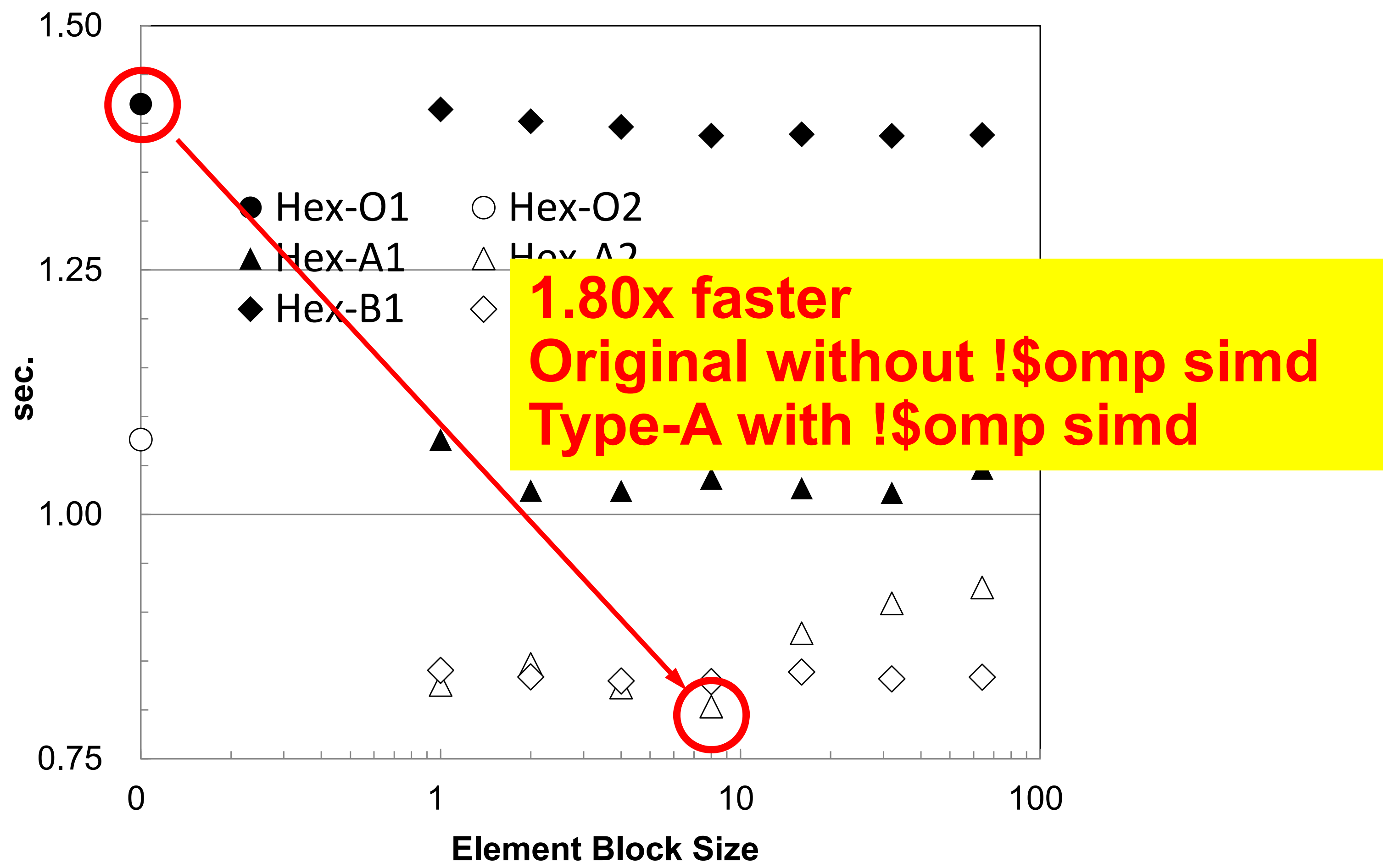
Significant Improvement by !$omp simd

Comp. Time for Matrix Assembly

KNC: 240 Threads x 1

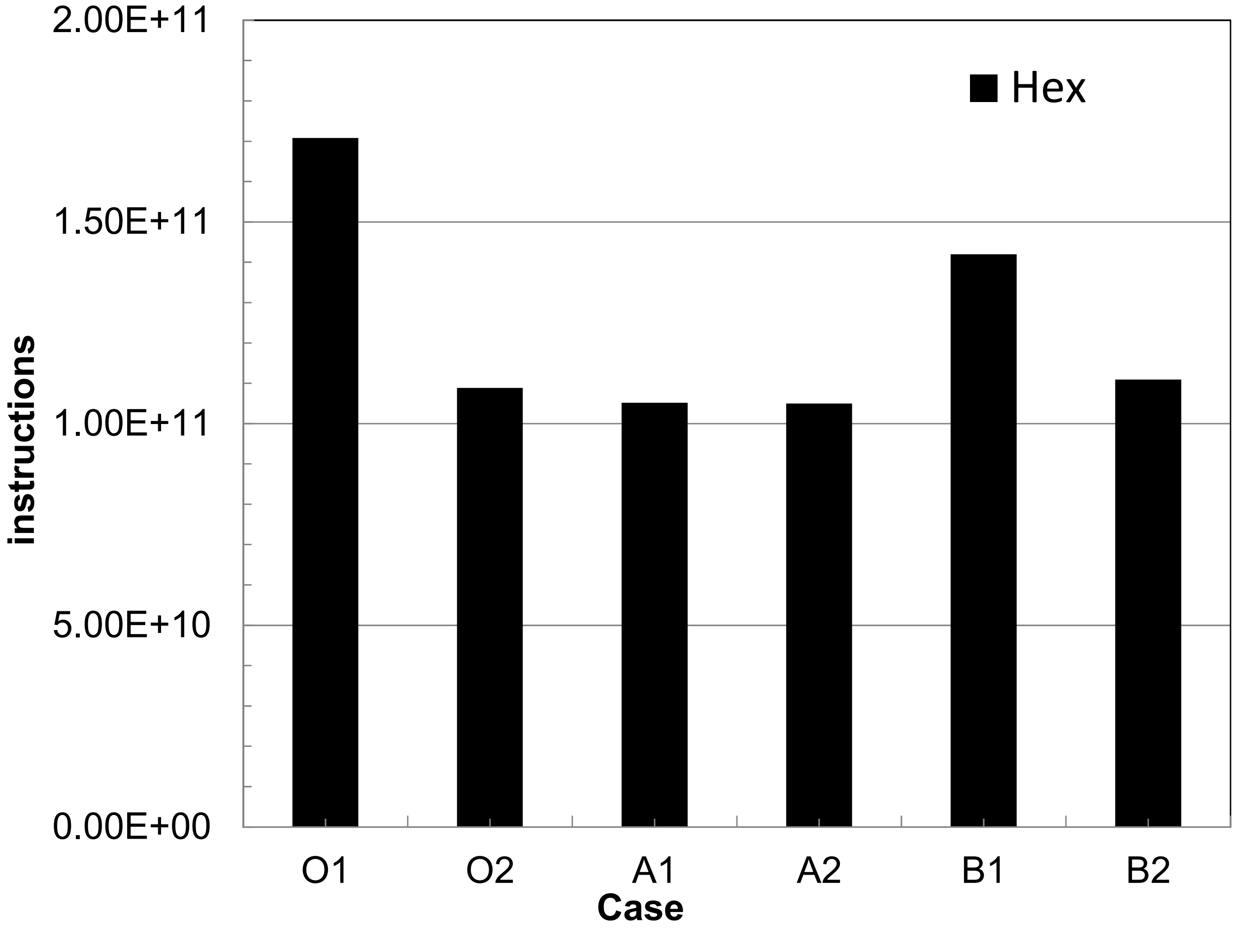
O: Original, A: Type-A, B: Type-B

1: without !$omp simd, 2: with !$omp simd



Instruction # for Matrix Assembly

KNC: 240 Threads x 1, Vtune
O: Original, A: Type-A, B: Type-B
1: without !$omp simd, 2: with !$omp simd



Summary

- Iterative Solvers
  - Format for Storing Sparse Matrices
    - MIC: Significant
    - IvyB: Very small (Out-of-Order architecture?, What’s happening in KNL ?)
  - Vectorization
    - Effects are not so clear in ICCG
    - Future Works: SELL-C-Sigma in ICCG
  - Compile Options
    - Interesting results
    - An automatic tool for experiments under development
- Matrix Assembly
  - Effects of vectorization is significant