

全国共同利用施設

東京大学情報基盤センター

Information Technology Center, The University of Tokyo



東京大学

THE UNIVERSITY OF TOKYO

# Iterative Linear Solvers for Sparse Matrices

**Kengo Nakajima**

Information Technology Center, The University of Tokyo, Japan

2013 International Summer School on HPC Challenges  
in Computational Sciences

New York University, New York, NY

June 24-28, 2013

# TOC

- Sparse Matrices
- Iterative Linear Solvers
  - Preconditioning
  - Parallel Iterative Linear Solvers
  - Multigrid Method
  - Recent Technical Issues
- Example of Parallel MGCG
- ppOpen-HPC

# TOC

- Sparse Matrices
- Iterative Linear Solvers
  - Preconditioning
  - Parallel Iterative Linear Solvers
  - **Multigrid Method**
  - **Recent Technical Issues**
- **Example of Parallel MGCG**
- ppOpen-HPC

# Parallel Iterative Solvers

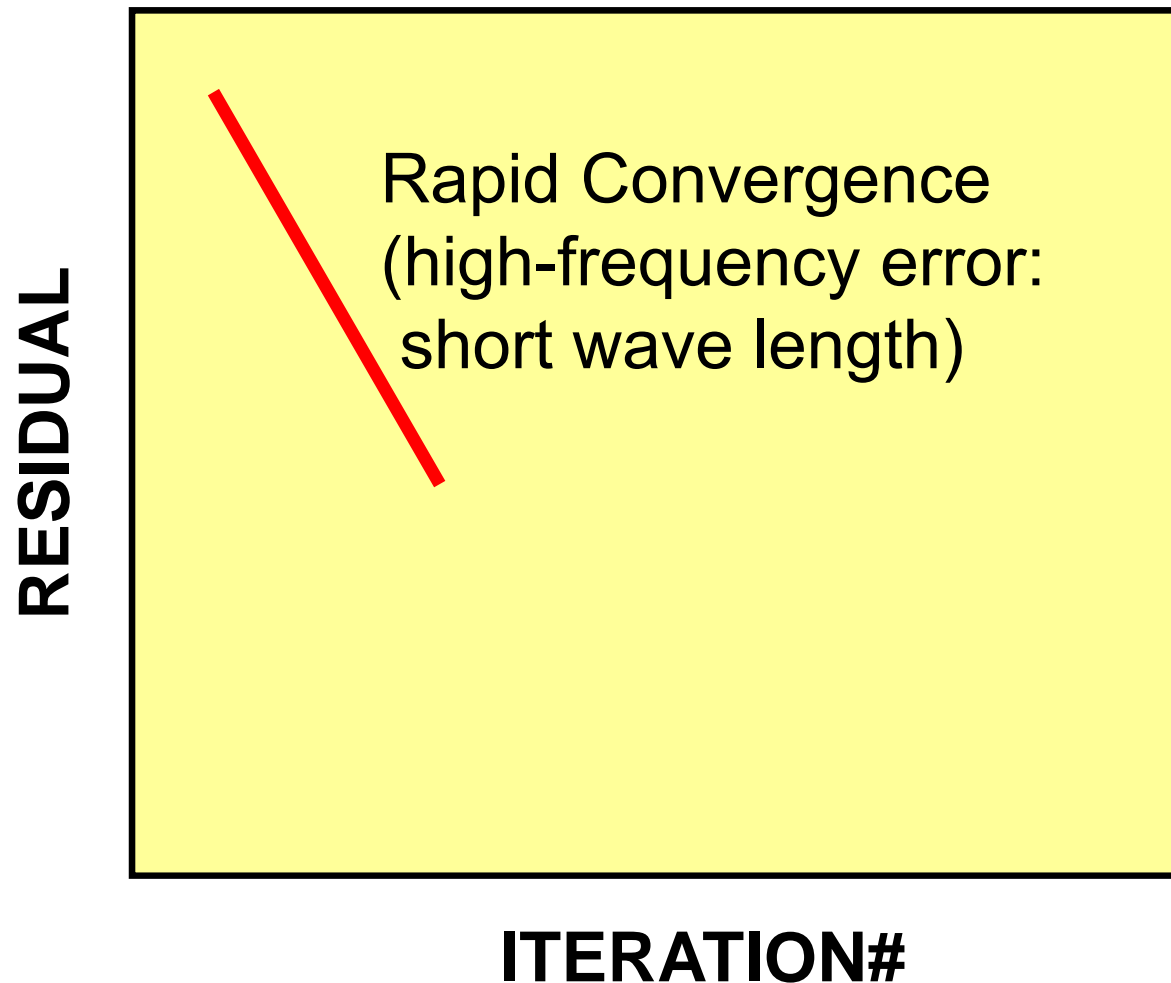
- Both of convergence (robustness) and efficiency (single/parallel) are important
- Global communications needed
  - Mat-Vec (P2P communications, MPI\_Isend/Irecv/Waitall): Local Data Structure with HALO
    - effect of latency
  - Dot-Products (MPI\_Allreduce)
  - Preconditioning (up to algorithm)
- Remedy for Robust Parallel ILU Preconditioner
  - Additive Schwarz Domain Decomposition
  - HID (Hierarchical Interface Decomposition, based on global nested dissection) [Henon & Saad 2007], ext. HID [KN 2010]
- Parallel “Direct” Solvers (e.g. SuperLU, MUMPS etc.)

- Sparse Matrices
- Iterative Linear Solvers
  - Preconditioning
  - Parallel Iterative Linear Solvers
  - Multigrid Method
  - Recent Technical Issues
- Example of Parallel MGCG
- ppOpen-HPC

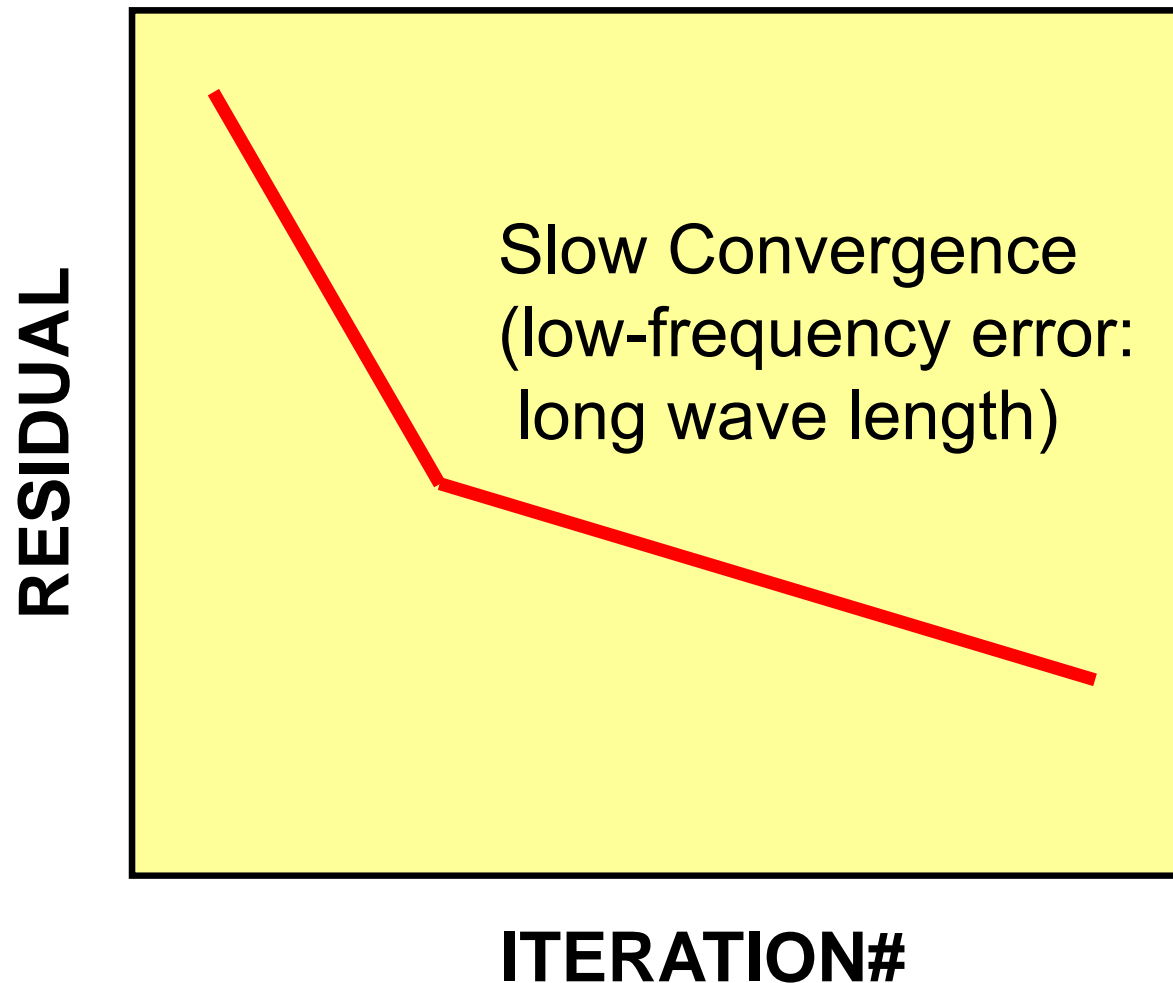
# Around the multigrid in a single slide

- Multigrid is a scalable method for solving linear equations.
- Relaxation methods (smoother/smoothing operator in MG world) such as Gauss-Seidel efficiently damp high-frequency error but do not eliminate low-frequency error.
- The multigrid approach was developed in recognition that this low-frequency error can be accurately and efficiently solved on a coarser grid.
- Multigrid method uniformly damps all frequencies of error components with a computational cost that depends only linearly on the problem size (=scalable).
  - Good for large-scale computations
- Multigrid is also a good preconditioning algorithm for Krylov iterative solvers.

# Convergence of Gauss-Seidel & SOR



# Convergence of Gauss-Seidel & SOR





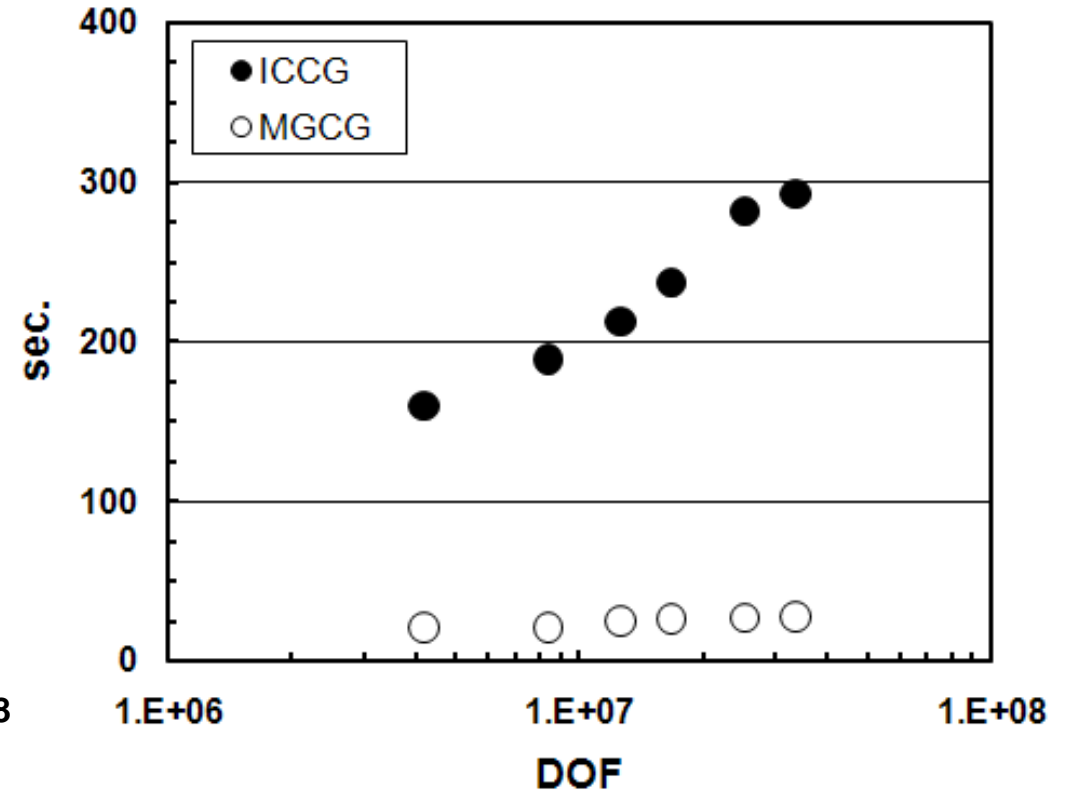
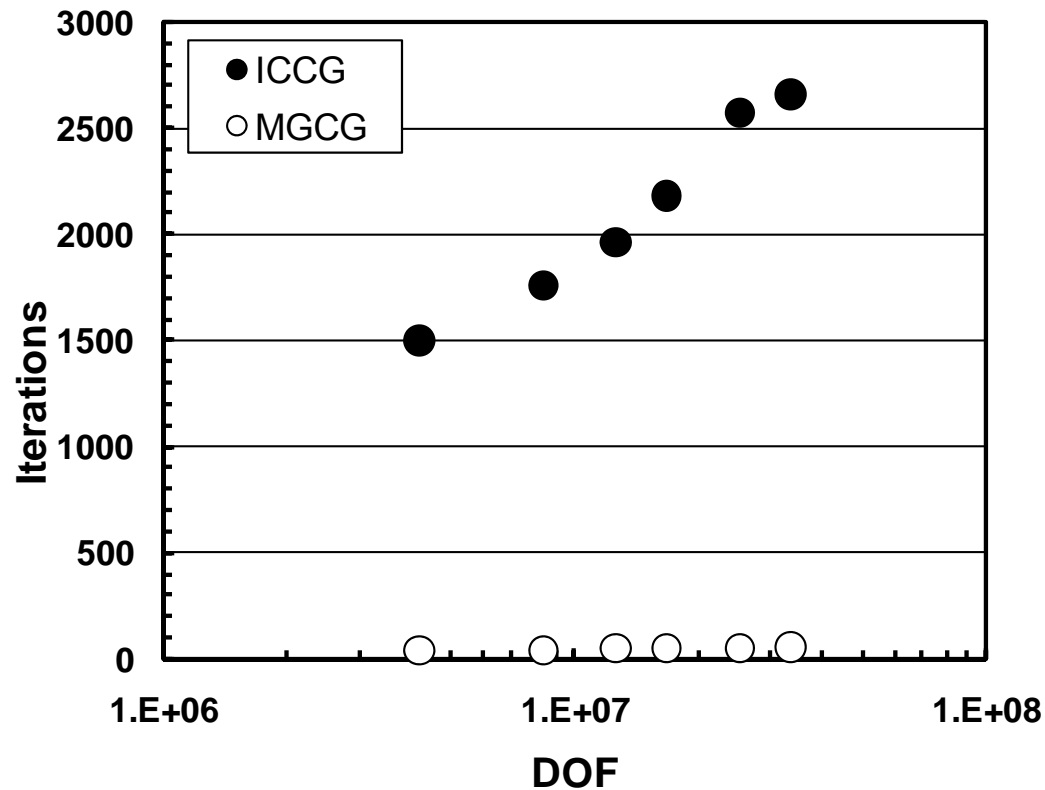
# Around the multigrid in a single slide

- Multigrid is a scalable method for solving linear equations.
- Relaxation methods (smoother/smoothing operator in MG world) such as Gauss-Seidel efficiently damp high-frequency error but do not eliminate low-frequency error.
- The multigrid approach was developed in recognition that this low-frequency error can be accurately and efficiently solved on a coarser grid.
- Multigrid method uniformly damps all frequencies of error components with a computational cost that depends only linearly on the problem size (=scalable).
  - Good for large-scale computations
- Multigrid is also a good preconditioning algorithm for Krylov iterative solvers.

# Multigrid is scalable

## Weak Scaling: Problem Size/Core Fixed for 3D Poisson Eqn's ( $\Delta\phi=q$ )

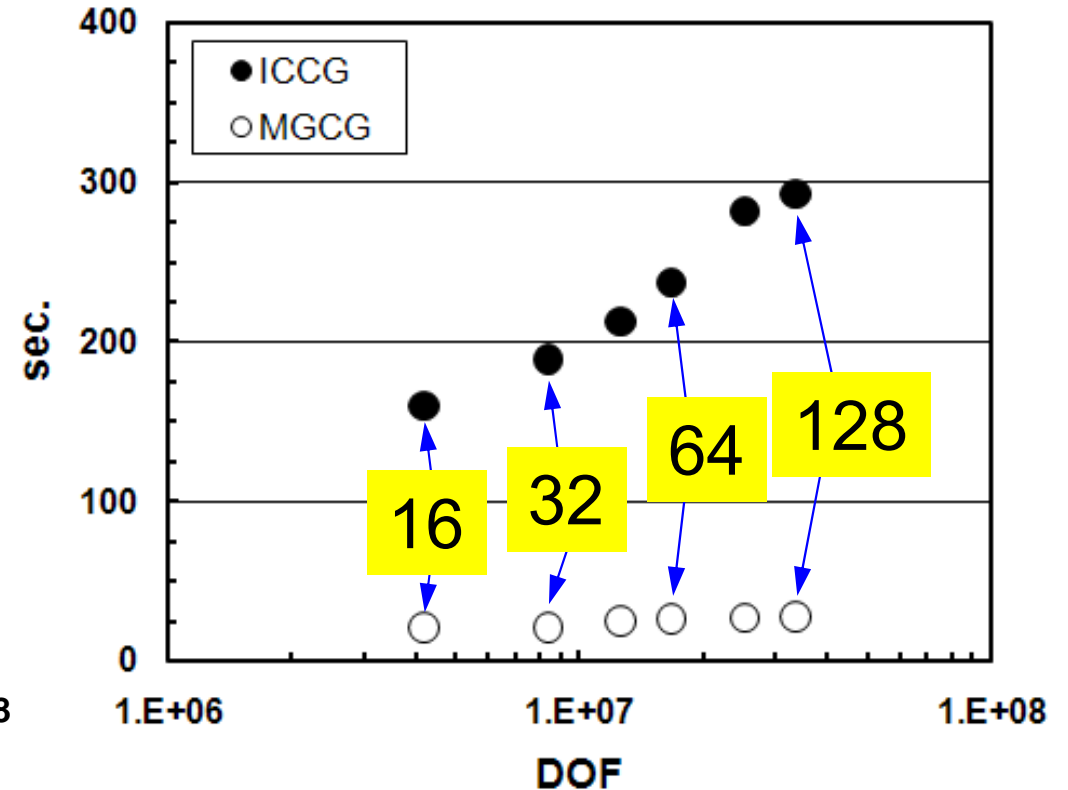
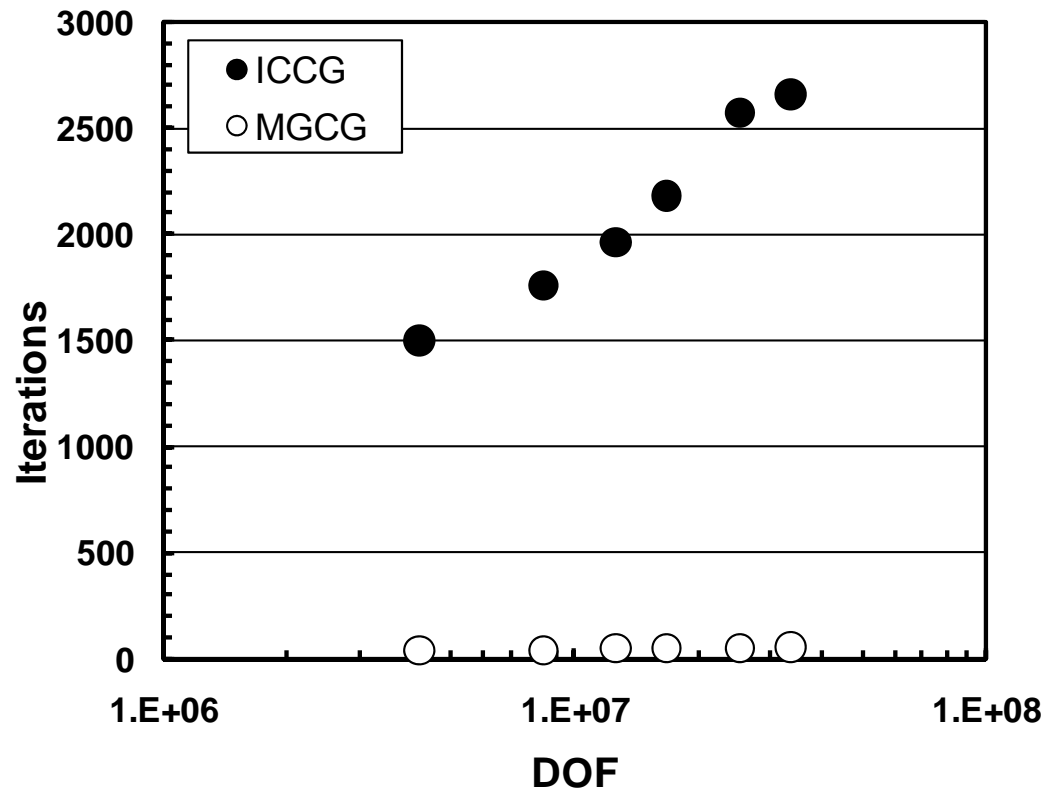
MGCG= Conjugate Gradient with Multigrid Preconditioning



# Multigrid is scalable

## Weak Scaling: Problem Size/Core Fixed

Comp. time of MGCG for weak scaling is constant:  
=> scalable



# Procedure of Multigrid (1/3)

Multigrid is a scalable method for solving linear equations. Relaxation methods such as Gauss-Seidel efficiently damp high-frequency error but do not eliminate low-frequency error. The multigrid approach was developed in recognition that this low-frequency error can be accurately and efficiently solved on a coarser grid. This concept is explained here in the following simple 2-level method. If we have obtained the following linear system on a fine grid :

$$A_F u_F = f$$

and  $A_C$  as the discrete form of the operator on the coarse grid, a simple coarse grid correction can be given by :

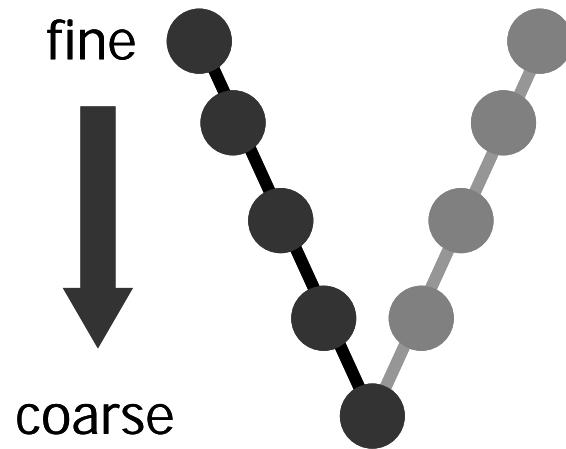
$$u_F^{(i+1)} = u_F^{(i)} + R^T A_C^{-1} R (f - A_F u_F^{(i)})$$

where  $R^T$  is the matrix representation of linear interpolation from the coarse grid to the fine grid (*prolongation* operator) and  $R$  is called the restriction operator. Thus, it is possible to calculate the residual on the fine grid, solve the coarse grid problem, and interpolate the coarse grid solution on the fine grid.

# Procedure of Multigrid (2/3)

This process can be described as follows :

1. Relax the equations on the fine grid and obtain the result  $u_F^{(i)} = S_F ( A_F, f )$ . This operator  $S_F$  (e.g., Gauss-Seidel) is called the *smoothing operator* (or ).
2. Calculate the residual term on the fine grid by  $r_F = f - A_F u_F^{(i)}$ .
3. *Restrict* the residual term on to the coarse grid by  $r_C = R r_F$ .
4. Solve the equation  $A_C u_C = r_C$  on the coarse grid ; the accuracy of the solution on the coarse grid affects the convergence of the entire multigrid system.
5. Interpolate (or *prolong*) the coarse grid correction on the fine grid by  $Du_C^{(i)} = R^T u_C$ .
6. Update the solution on the fine grid by  $u_F^{(i+1)} = u_F^{(i)} + Du_C^{(i)}$



$$\mathbf{L}^k \mathbf{W}^k = \mathbf{F}^k \quad (\text{Linear Equation: Fine Level})$$

$$\mathbf{R}^k = \mathbf{F}^k - \mathbf{L}^k \mathbf{w}_1^k$$

$$\mathbf{v}^k = \mathbf{W}^k - \mathbf{w}_1^k, \mathbf{L}^k \mathbf{v}^k = \mathbf{R}^k$$

$$\mathbf{R}^{k-1} = \mathbf{I}_k^{k-1} \mathbf{R}^k$$

$$\mathbf{L}^{k-1} \mathbf{v}^{k-1} = \mathbf{R}^{k-1} \quad (\text{Linear Equation: Coarse Level})$$

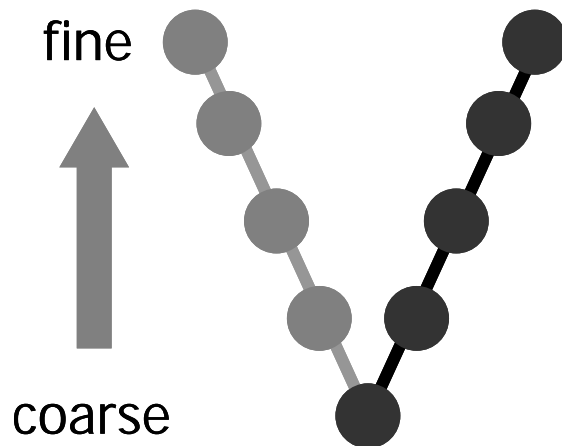
$$\mathbf{v}^k = \mathbf{I}_{k-1}^k \mathbf{v}^{k-1}$$

$$\mathbf{w}_2^k = \mathbf{w}_1^k + \mathbf{v}^k$$

$\mathbf{w}_1^k$  : Approx. Solution

$\mathbf{v}^k$  : Correction

$\mathbf{I}_k^{k-1}$  : Restriction Operator



$$\mathbf{L}^k \mathbf{W}^k = \mathbf{F}^k \quad (\text{Linear Equation: Fine Level})$$

$$\mathbf{R}^k = \mathbf{F}^k - \mathbf{L}^k \mathbf{w}_1^k$$

$$\mathbf{v}^k = \mathbf{W}^k - \mathbf{w}_1^k, \mathbf{L}^k \mathbf{v}^k = \mathbf{R}^k$$

$$\mathbf{R}^{k-1} = \mathbf{I}_k^{k-1} \mathbf{R}^k$$

$$\mathbf{L}^{k-1} \mathbf{v}^{k-1} = \mathbf{R}^{k-1} \quad (\text{Linear Equation: Coarse Level})$$

$$\mathbf{v}^k = \mathbf{I}_{k-1}^k \mathbf{v}^{k-1}$$

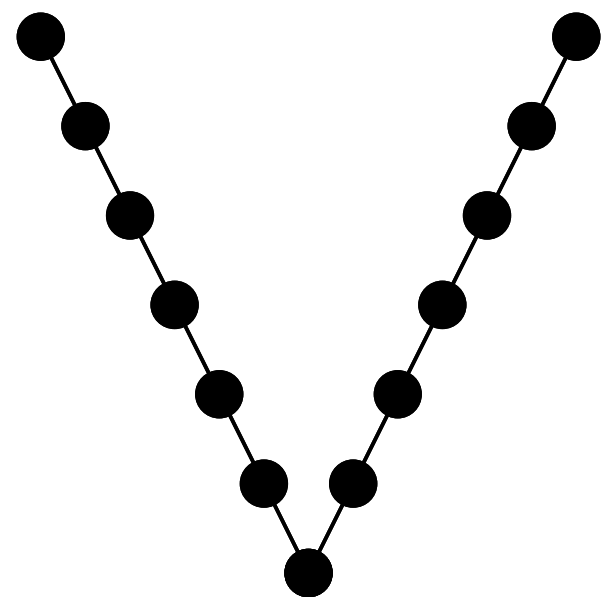
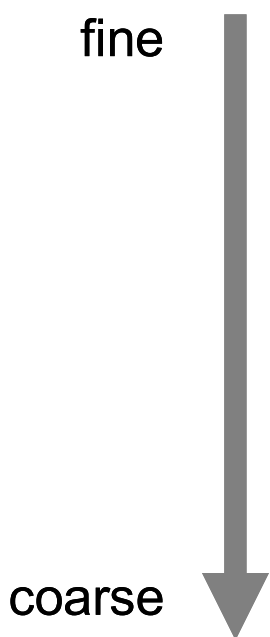
$$\mathbf{w}_2^k = \mathbf{w}_1^k + \mathbf{v}^k$$

$\mathbf{I}_{k-1}^k$  : Prolongation Operator

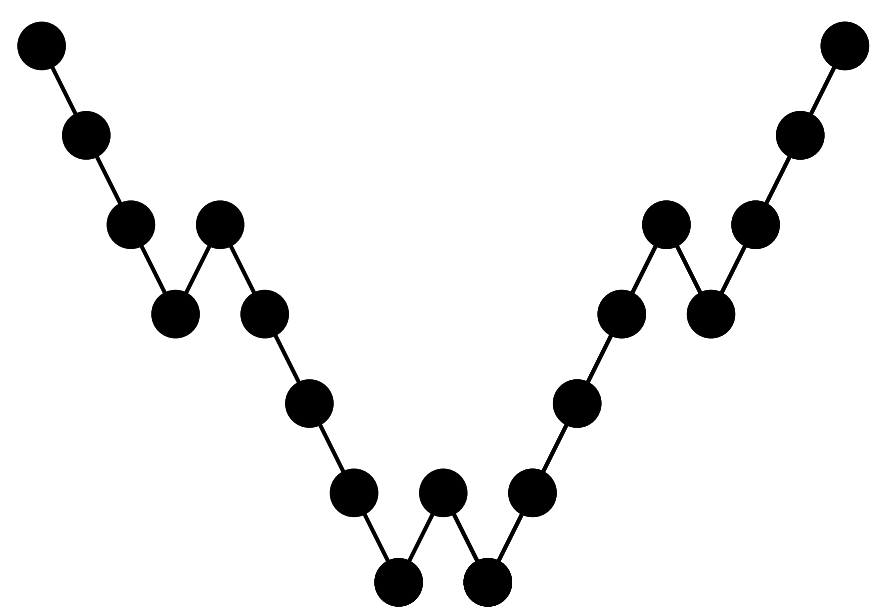
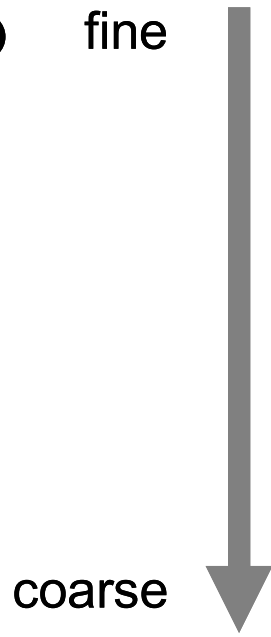
$\mathbf{w}_2^k$  : Approx. Solution by Multigrid

# Procedure of Multigrid (3/3)

- Recursive application of this algorithm for 2-level procedure to consecutive systems of coarse-grid equations gives a multigrid V-cycle. If the components of the V-cycle are defined appropriately, the result is a method that uniformly damps all frequencies of error with a computational cost that depends only linearly on the problem size.
  - In other words, multigrid algorithms are *scalable*.
- In the V-cycle, starting with the finest grid, all subsequent coarser grids are visited only once.
  - In the down-cycle, smoothers damp oscillatory error components at different grid scales.
  - In the up-cycle, the smooth error components remaining on each grid level are corrected using the error approximations on the coarser grids.
- Alternatively, in a W-cycle, the coarser grids are solved more rigorously in order to reduce residuals as much as possible before going back to the more expensive finer grids.



(a) V-Cycle



(b) W-Cycle



# Multigrid as a Preconditioner

- Multigrid algorithms tend to be problem-specific solutions and less robust than preconditioned Krylov iterative methods such as the IC/ILU methods.
- Fortunately, it is easy to combine the best features of multigrid and Krylov iterative methods into one algorithm
  - multigrid-preconditioned Krylov iterative methods.
- The resulting algorithm is robust, efficient and scalable.
- Multigrid solvers and Krylov iterative solvers preconditioned by multigrid are intrinsically suitable for parallel computing.

# Geometric and Algebraic Multigrid

- One of the most important issues in multigrid is the construction of the coarse grids.
- There are 2 basic multigrid approaches
  - geometric and algebraic
- In geometric multigrid, the geometry of the problem is used to define the various multigrid components.
- In contrast, algebraic multigrid methods use only the information available in the linear system of equations, such as matrix connectivity.
- Algebraic multigrid method (AMG) is suitable for applications with unstructured grids.
- Many tools for both geometric and algebraic methods on unstructured grids have been developed.

# “Dark Side” of Multigrid Method

- Its performance is excellent for well-conditioned simple problems, such as homogeneous Poisson equations.
- But convergence could be worse for ill-conditioned problems.
- Extension of applicability of multigrid method is an active research area.

# References

- Briggs, W.L., Henson, V.E. and McCormick, S.F. (2000)  
A Multigrid Tutorial Second Edition, SIAM
- Trottemberg, U., Oosterlee, C. and Schüller, A. (2001)  
Multigrid, Academic Press
- <https://computation.llnl.gov/casc/>
- Hypre (AMG Library)
  - [https://computation.llnl.gov/casc/linear\\_solvers/sls\\_hypre.html](https://computation.llnl.gov/casc/linear_solvers/sls_hypre.html)

- Sparse Matrices
- Iterative Linear Solvers
  - Preconditioning
  - Parallel Iterative Linear Solvers
  - Multigrid Method
  - Recent Technical Issues
- Example of Parallel MGCG
- ppOpen-HPC

# Key-Issues for Appl's/Algorithms towards Post-Peta & Exa Computing

Jack Dongarra (ORNL/U. Tennessee) at ISC 2013

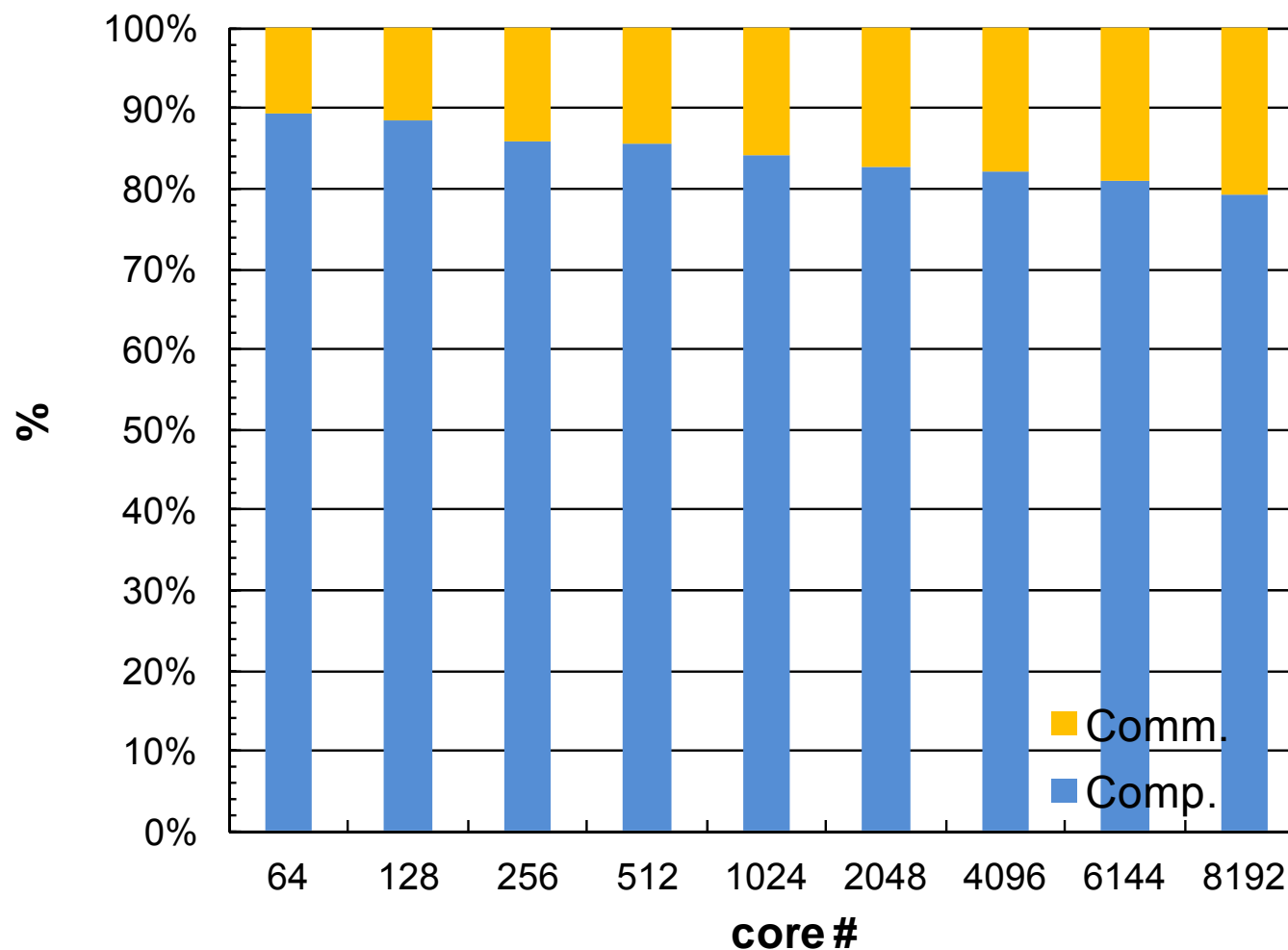
- Hybrid/Heterogeneous Architecture
  - Multicore + GPU/Manycores (Intel MIC/Xeon Phi)
    - Data Movement, Hierarchy of Memory
- Communication/Synchronization Reducing Algorithms
- Mixed Precision Computation
- Auto-Tuning/Self-Adapting
- Fault Resilient Algorithms
- Reproducibility of Results

# Recent Technical Issues in Parallel Iterative Solvers

- **Communication overhead becomes significant**
- Communication-Computation Overlap
  - Not so effective for Mat-Vec operations
- Communication Avoiding/Reducing Algorithms
- OpenMP/MPI Hybrid Parallel Programming Model
  - (Next section)

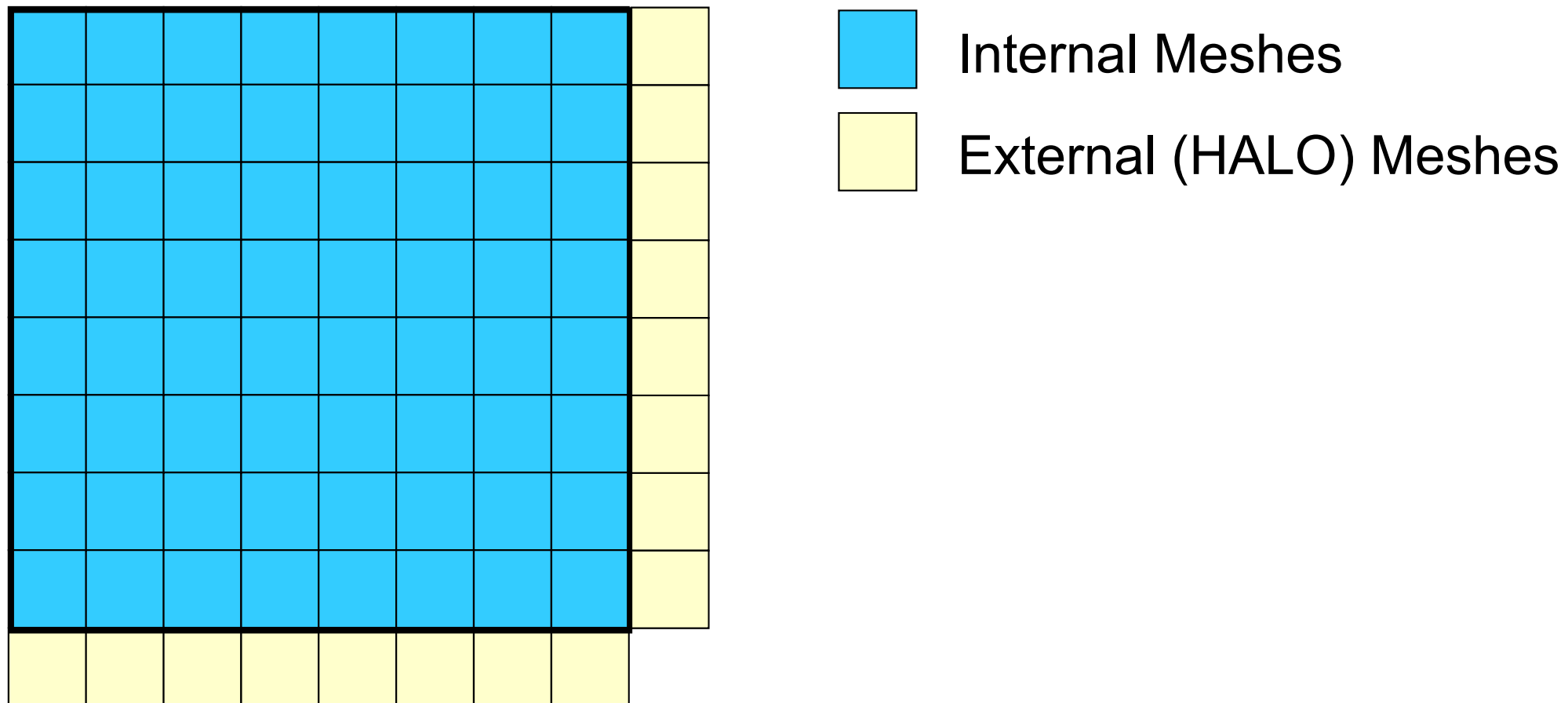
# Communication overhead becomes larger as node/core number increases

## Weak Scaling: MGCG on T2K Tokyo

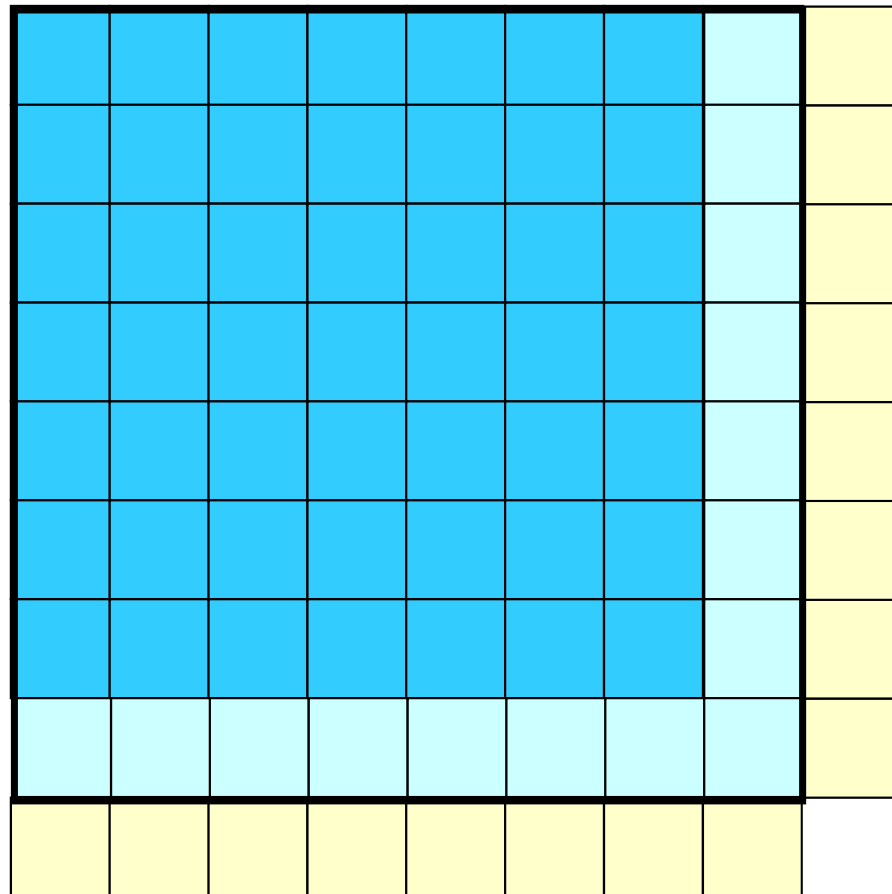



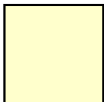
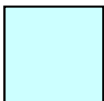


# Comm.-Comp. Overlapping



# Comm.-Comp. Overlapping



-  Internal Meshes
-  External (HALO) Meshes
-  Internal Meshes on Boundary's

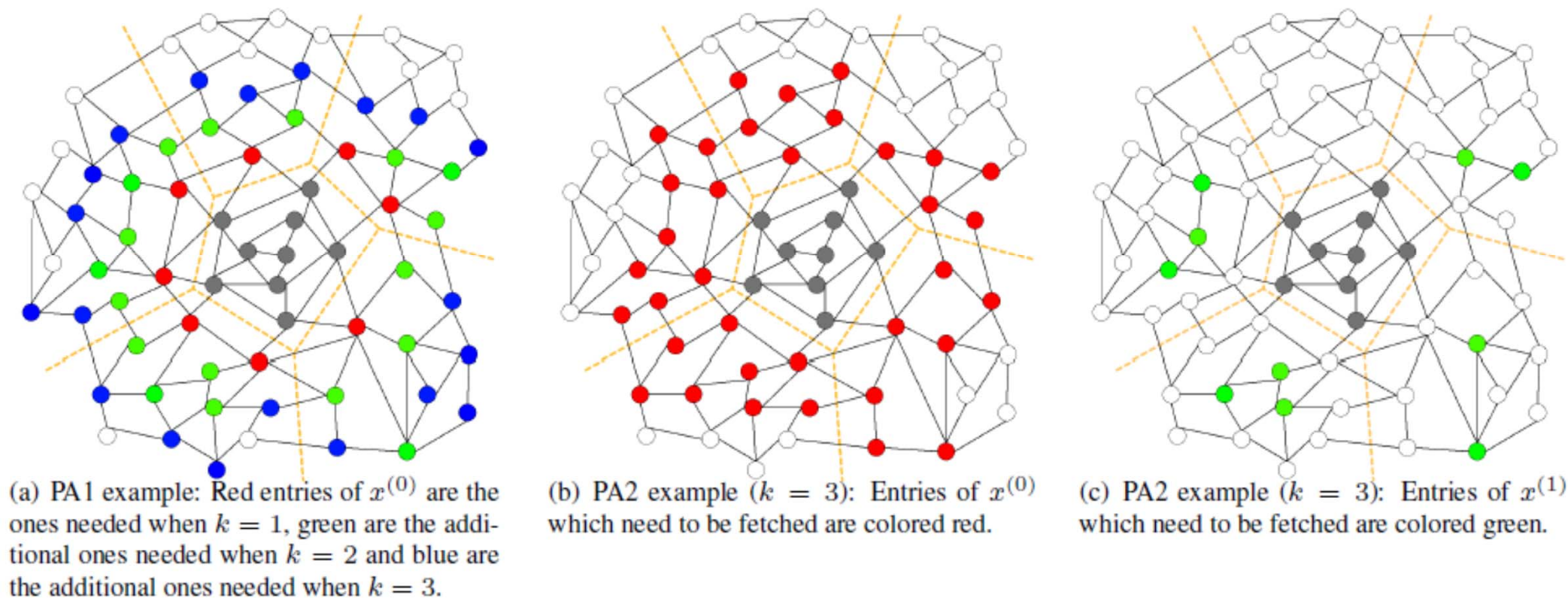
## Mat-Vec operations

- Overlapping of computations of internal meshes, and importing external meshes.
- Then computation of international meshes on boundary's
- Difficult for IC/ILU on Hybrid

# Communication Avoiding/Reducing Algorithms for Sparse Linear Solvers

- Krylov Iterative Method without Preconditioning
  - Demmel, Hoemmen, Mohiyuddin etc. (UC Berkeley)
- *s-step* method
  - Just one P2P communication for each Mat-Vec during  $s$  iterations. Convergence becomes unstable for large  $s$ .
  - matrix powers kernel:  $Ax$ ,  $A^2x$ ,  $A^3x$  ...
    - additional computations needed
- Communication Avoiding ILU0 (CA-ILU0) [Moufawad & Grigori, 2013]
  - First attempt to CA preconditioning
  - Nested dissection reordering for limited geometries (2D FDM)

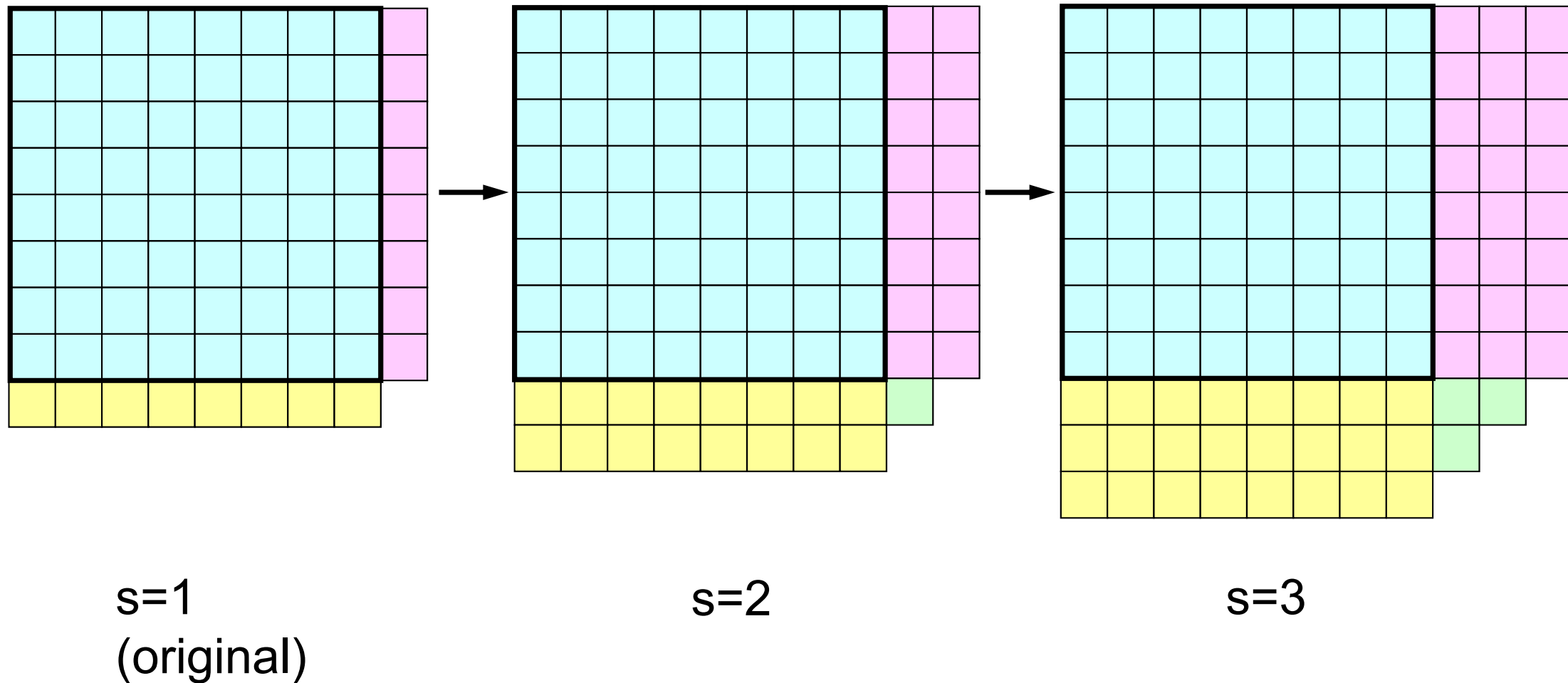
# Comm. Avoiding Krylov Iterative Methods using “Matrix Powers Kernel”



**Figure 2. Example for PA1 and PA2. The dotted lines define the different blocks. Each block resides on a different processor. The example shows from the perspective of the processor holding the central block.**

*Avoiding Communication in Sparse Matrix Computations.*  
James Demmel, Mark Hoemmen, Marghoob Mohiyuddin,  
and Katherine Yelick. , 2008 IPDPS

# Required Information of Local Meshes for $s$ -step CA computations (2D 5pt.)



- Sparse Matrices
- Iterative Linear Solvers
  - Preconditioning
  - Parallel Iterative Linear Solvers
  - Multigrid Method
  - Recent Technical Issues
- Example of Parallel MGCG
- ppOpen-HPC

# Key-Issues for Appl's/Algorithms towards Post-Peta & Exa Computing

Jack Dongarra (ORNL/U. Tennessee) at ISC 2013

- Hybrid/Heterogeneous Architecture
  - Multicore + GPU/Manycores (Intel MIC/Xeon Phi)
    - Data Movement, Hierarchy of Memory
- Communication/Synchronization Reducing Algorithms
- Mixed Precision Computation
- Auto-Tuning/Self-Adapting
- Fault Resilient Algorithms
- Reproducibility of Results

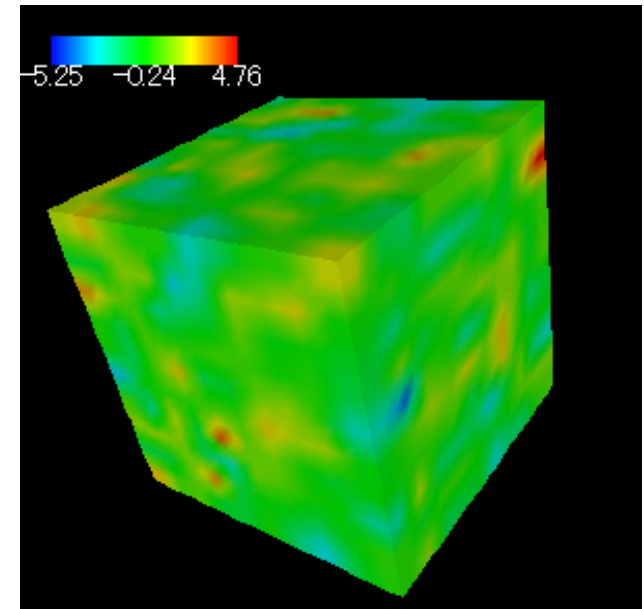
# Motivation of This Study

- Large-scale 3D Groundwater Flow
  - Poisson equations
  - Heterogeneous porous media
- Parallel (Geometric) Multigrid Solvers for FVM-type appl. on Fujitsu PRIMEHPC FX10 at University of Tokyo (Oakleaf-FX)
- Flat MPI vs. Hybrid (OpenMP+MPI)
- Expectations for Hybrid Parallel Programming Model
  - Number of MPI processes (and sub-domains) to be reduced
  - $O(10^8-10^9)$ -way MPI might not scale in Exascale Systems
  - Easily extended to Heterogeneous Architectures
    - CPU+GPU, CPU+Manycores (e.g. Intel MIC/Xeon Phi)
    - MPI+X: OpenMP, OpenACC, CUDA, OpenCL



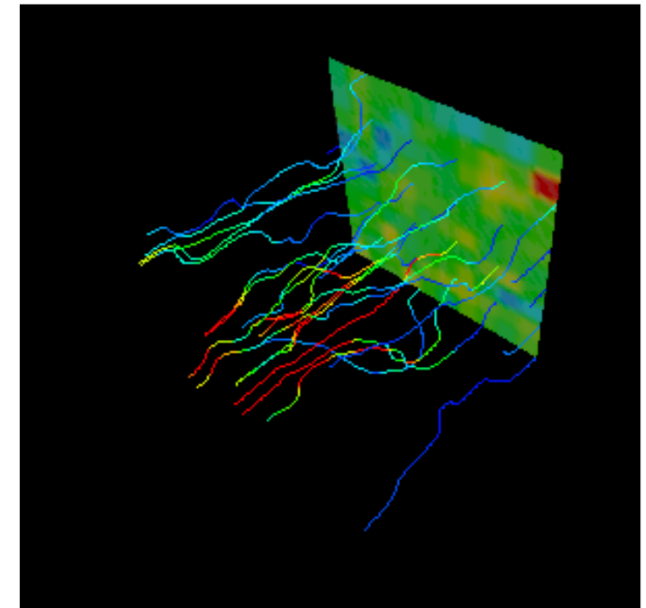
# Target Application: *pGW3D-FVM*

- 3D Groundwater Flow via. Heterogeneous Porous Media
  - Poisson's equation
  - Randomly distributed water conductivity
$$\nabla \cdot (\lambda(x, y, z) \nabla \phi) = q, \phi = 0 \text{ at } z = z_{\max}$$
  - Distribution of water conductivity is defined through methods in geostatistics [Deutsch & Journel, 1998]
- Finite-Volume Method on Cubic Voxel Mesh
- Distribution of Water Conductivity
  - $10^{-5}$ - $10^{+5}$ , Condition Number  $\sim 10^{+10}$
  - Average: 1.0
- Cyclic Distribution:  $128^3$



# Target Application: *pGW3D-FVM*

- 3D Groundwater Flow via. Heterogeneous Porous Media
  - Poisson's equation
  - Randomly distributed water conductivity
$$\nabla \cdot (\lambda(x, y, z) \nabla \phi) = q, \phi = 0 \text{ at } z = z_{\max}$$
  - Distribution of water conductivity is defined through methods in geostatistics [Deutsch & Journel, 1998]
- Finite-Volume Method on Cubic Voxel Mesh
- Distribution of Water Conductivity
  - $10^{-5}$ - $10^{+5}$ , Condition Number  $\sim 10^{+10}$
  - Average: 1.0
- Cyclic Distribution:  $128^3$



# Motivation of This Study

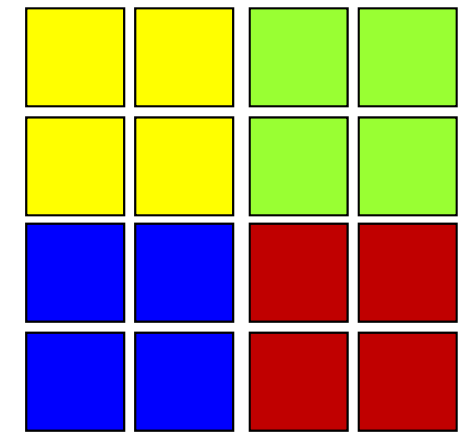
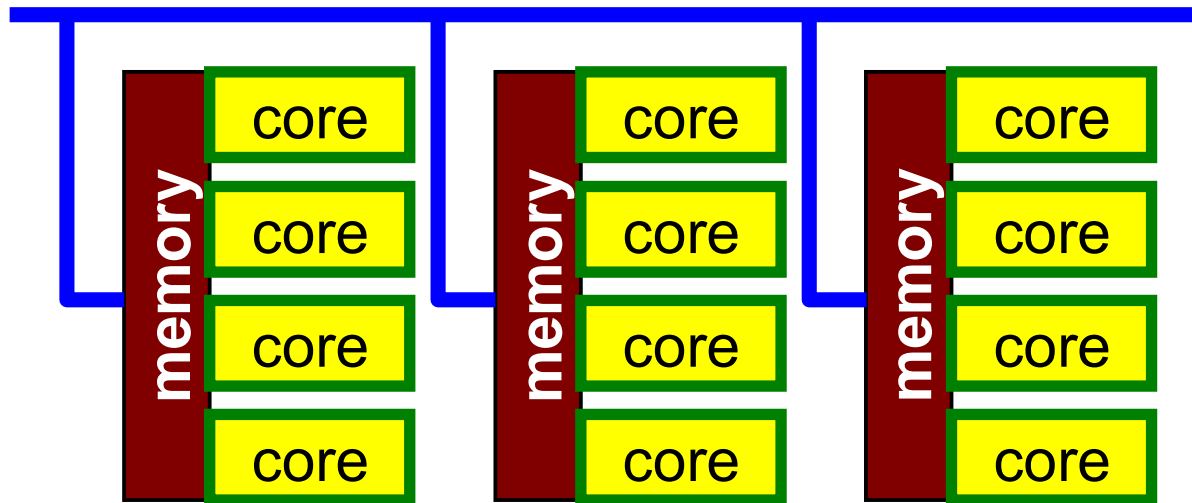
- Large-scale 3D Groundwater Flow
  - Poisson equations
  - Heterogeneous porous media
- Parallel (Geometric) Multigrid Solvers for FVM-type appl. on Fujitsu PRIMEHPC FX10 at University of Tokyo (Oakleaf-FX)
- Flat MPI vs. Hybrid (OpenMP+MPI)
- Expectations for Hybrid Parallel Programming Model
  - Number of MPI processes (and sub-domains) to be reduced
  - $O(10^8-10^9)$ -way MPI might not scale in Exascale Systems
  - Easily extended to Heterogeneous Architectures
    - CPU+GPU, CPU+Manycorers (e.g. Intel MIC/Xeon Phi)
    - MPI+X: OpenMP, OpenACC, CUDA, OpenCL

# Keywords

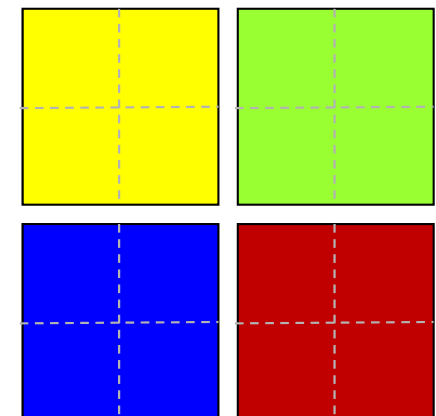
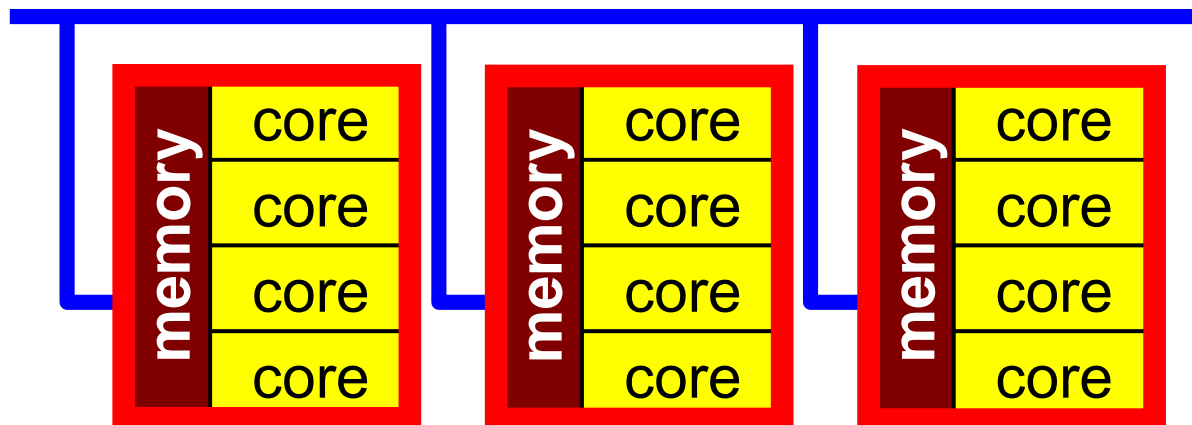
- Parallel Geometric Multigrid
- OpenMP/MPI Hybrid Parallel Programming Model
- Localized Block Jacobi Preconditioning
  - Overlapped Additive Schwartz Domain Decomposition (ASDD)
- OpenMP Parallelization with Coloring
- Coarse Grid Aggregation (CGA), Hierarchical CGA

# Flat MPI vs. Hybrid

Flat-MPI: Each Core -> Independent

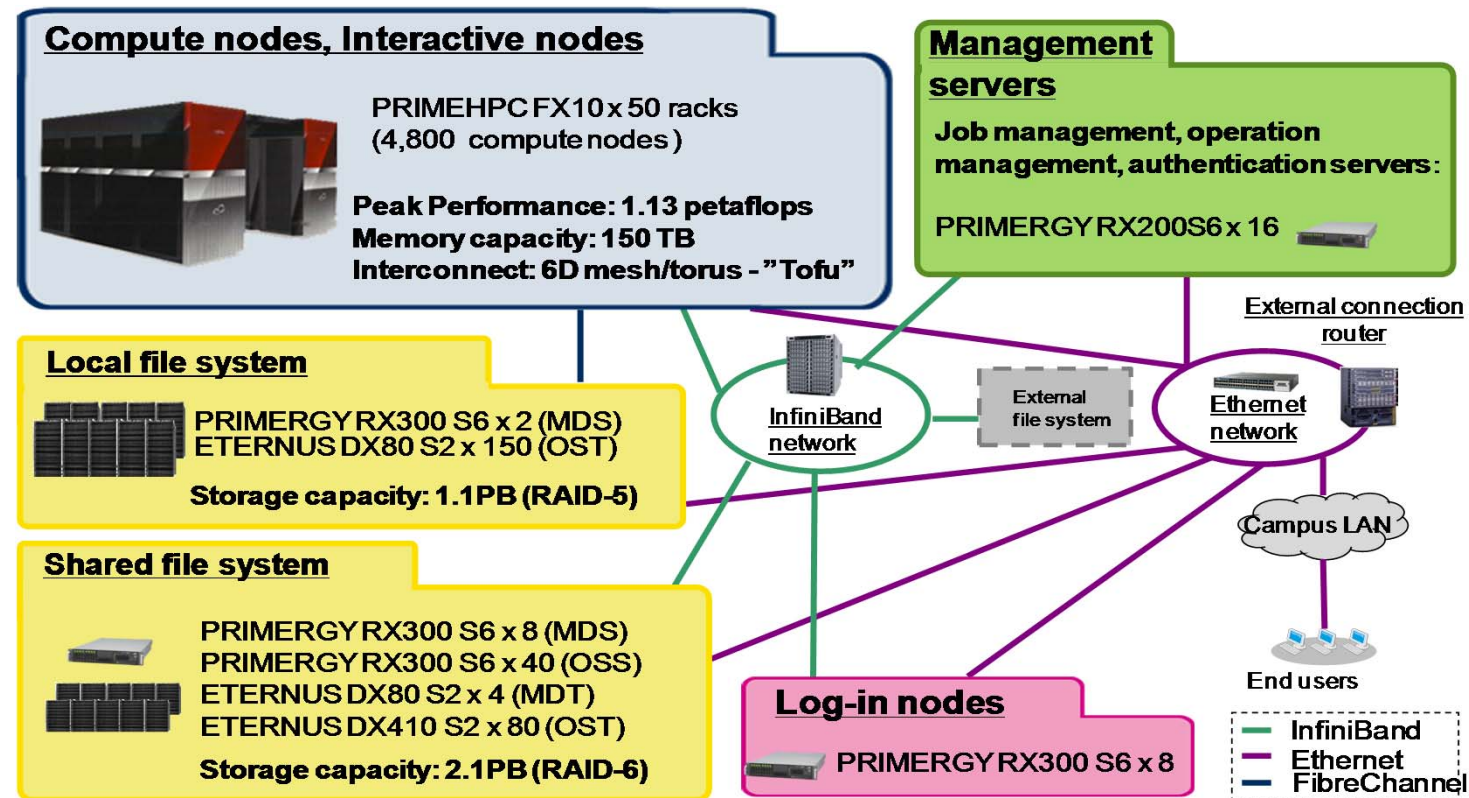


Hybrid: Hierarchical Structure



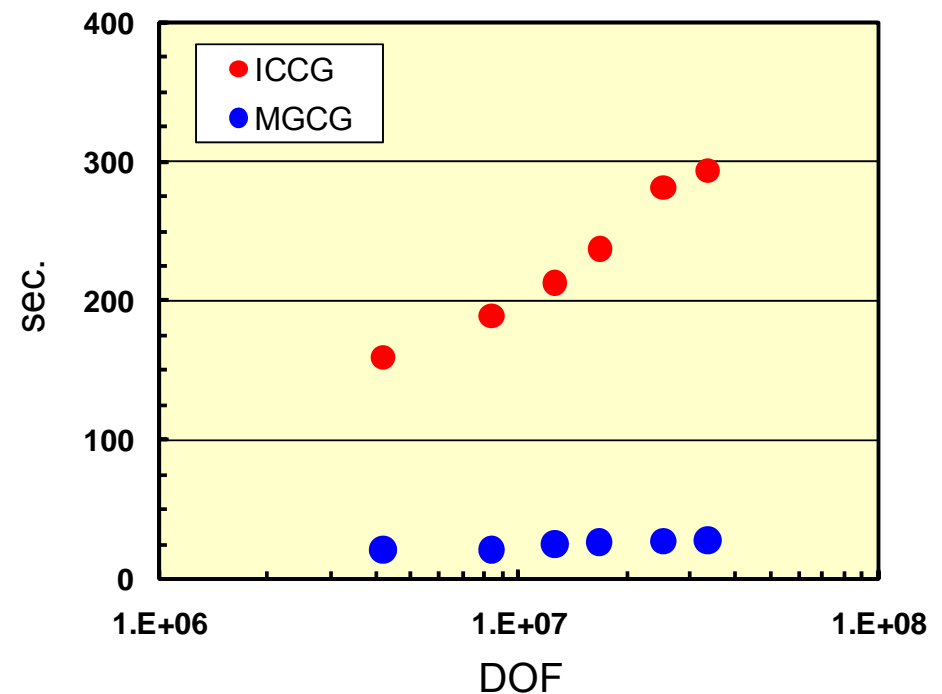
# Fujitsu PRIMEHPC FX10 (Oakleaf-FX) at the U. Tokyo

- SPARC64 lxfx (4,800 nodes, 76,800 cores)
- Commercial version of K computerx
- Peak: 1.13 PFLOPS (1.043 PF, 26<sup>th</sup>, 41<sup>th</sup> TOP 500 in 2013 June.)
- Memory BWTH 398 TB/sec.



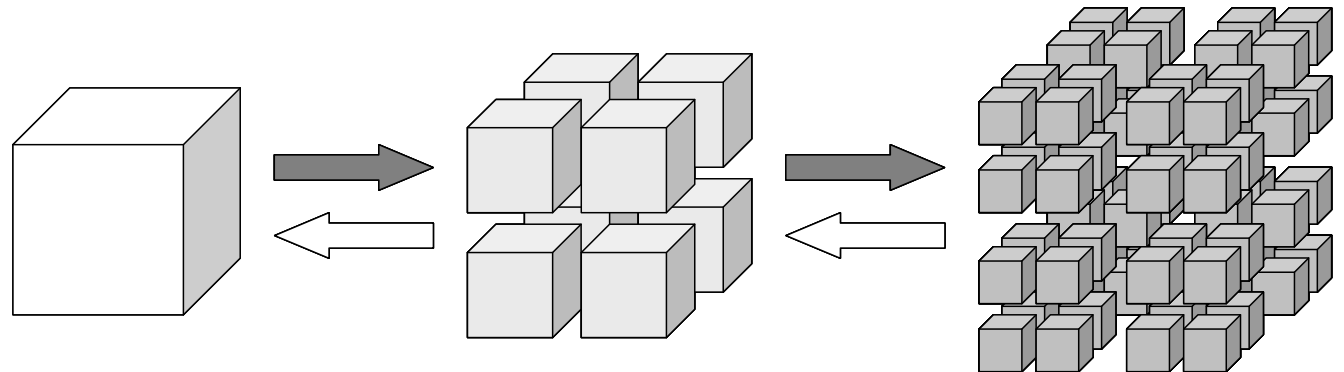
# Multigrid

- Scalable Multi-Level Method using Multilevel Grid for Solving Linear Eqn's
  - Computation Time  $\sim O(N)$  (N: # unknowns)
  - Good for large-scale problems
- Preconditioner for Krylov Iterative Linear Solvers
  - MGCG



# Linear Solvers

- Preconditioned CG Method
  - Multigrid Preconditioning (MGCG)
  - IC(0) for Smoothing Operator (Smoother): good for ill-conditioned problems
- Parallel Geometric Multigrid Method
  - 8 fine meshes (children) form 1 coarse mesh (parent) in isotropic manner (octree)
  - V-cycle
  - Domain-Decomposition-based: Localized Block-Jacobi, Overlapped Additive Schwarz Domain Decomposition (ASDD)
  - Operations using a single core at the coarsest level (redundant)



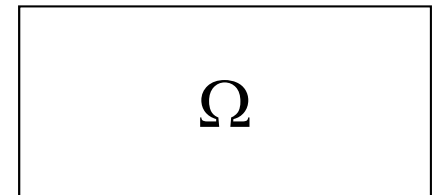


# Overlapped Additive Schwartz Domain Decomposition Method

ASDD: Localized Block-Jacobi Precond. is stabilized

## Global Operation

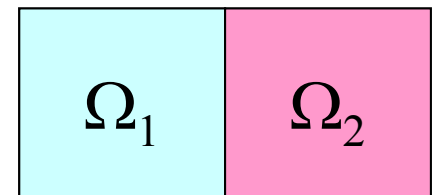
$$Mz = r$$



## Local Operation

$$z_{\Omega_1} = M_{\Omega_1}^{-1} r_{\Omega_1}, \quad z_{\Omega_2} = M_{\Omega_2}^{-1} r_{\Omega_2}$$

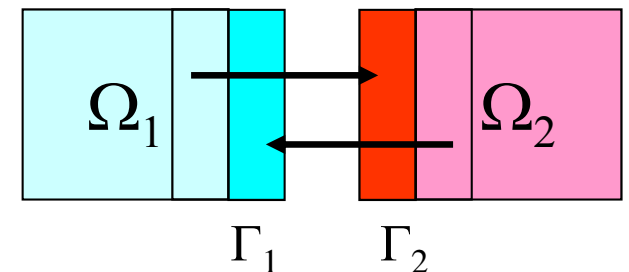
$\Omega_i$ : Internal ( $i \leq N$ )  
 $\Gamma_i$ : External ( $i > N$ )



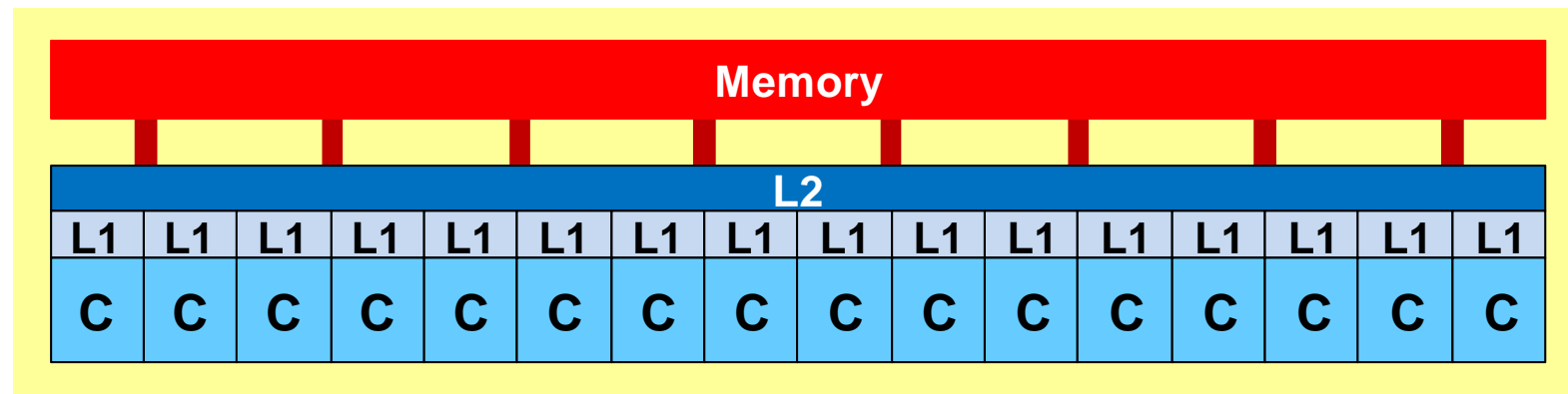
## Global Nesting Correction

$$z_{\Omega_1}^n = z_{\Omega_1}^{n-1} + M_{\Omega_1}^{-1} (r_{\Omega_1} - M_{\Omega_1} z_{\Omega_1}^{n-1} - M_{\Gamma_1} z_{\Gamma_1}^{n-1})$$

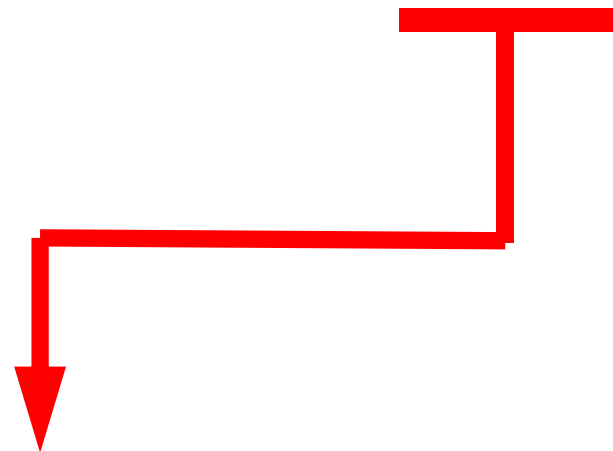
$$z_{\Omega_2}^n = z_{\Omega_2}^{n-1} + M_{\Omega_2}^{-1} (r_{\Omega_2} - M_{\Omega_2} z_{\Omega_2}^{n-1} - M_{\Gamma_2} z_{\Gamma_2}^{n-1})$$



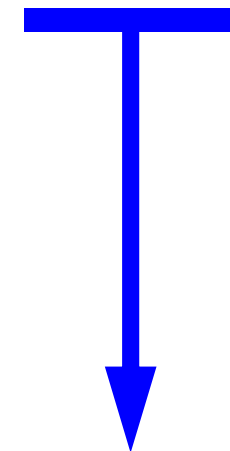




**HB M x N**



Number of OpenMP threads  
per a single MPI process



Number of MPI process  
per a single node

# Reordering for extracting parallelism in each domain (= MPI Process)

- Krylov Iterative Solvers
  - Dot Products
  - SMVP
  - DAXPY
  - **Preconditioning**
- IC/ILU Factorization, Forward/Backward Substitution
  - Global Data Dependency
  - Reordering needed for parallelism ([KN 2003] on the Earth Simulator, KN@CMCIM-2002)
  - Multicoloring, RCM, CM-RCM

# Parallelization of ICCG

## IC Factorization

```
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
```

## Forward Substitution

```
do i= 1, N
  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
```

# (Global) Data Dependency:

Writing/reading may occur simultaneously, hard to parallelize

## IC Factorization

```
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
```

## Forward Substitution

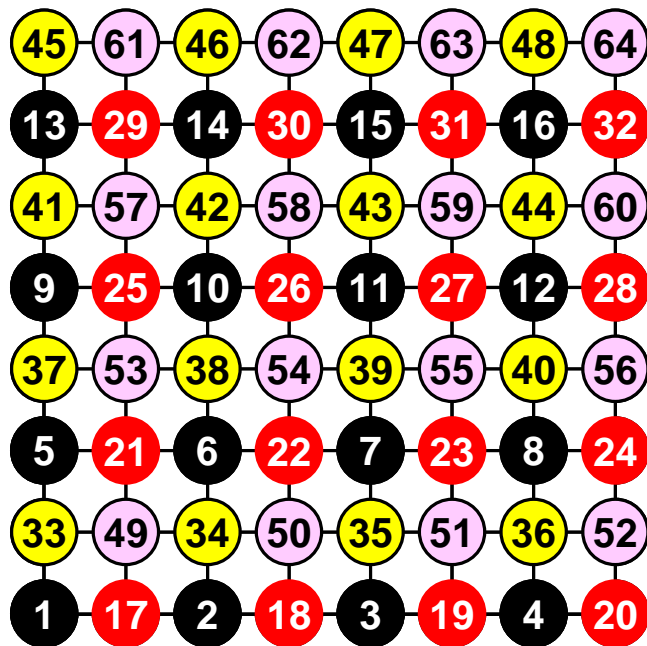
```
do i= 1, N
  WVAL= W(i, Z)
  do k= indexL(i-1)+1, indexL(i)
    WVAL= WVAL - AL(k) * W(itemL(k), Z)
  enddo
  W(i, Z) = WVAL * W(i, DD)
enddo
```

# OpenMP for SpMV: Straightforward NO data dependency

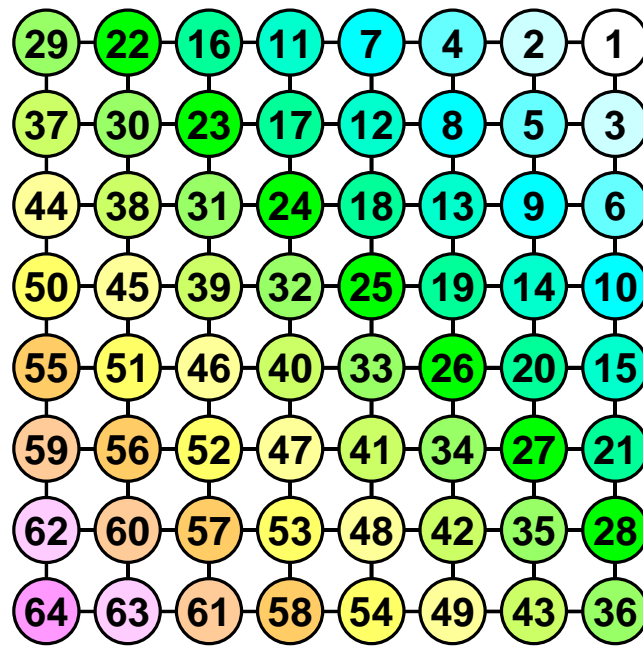
```
!$omp parallel do private(ip, i, VAL, k)
  do ip= 1, PEsmptOT
    do i = INDEX(ip-1)+1, INDEX(ip)
      VAL= D(i)*W(i, P)
      do k= indexL(i-1)+1, indexL(i)
        VAL= VAL + AL(k)*W(itemL(k), P)
      enddo
      do k= indexU(i-1)+1, indexU(i)
        VAL= VAL + AU(k)*W(itemU(k), P)
      enddo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

# Ordering Methods

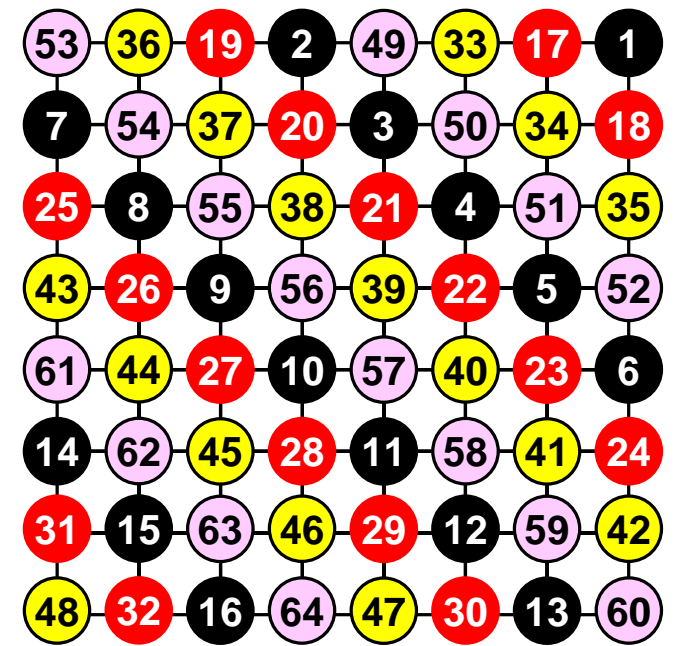
Elements in “same color” are independent: to be parallelized



**MC (Color#=4)**  
Multicoloring



**RCM**  
Reverse Cuthill-McKee

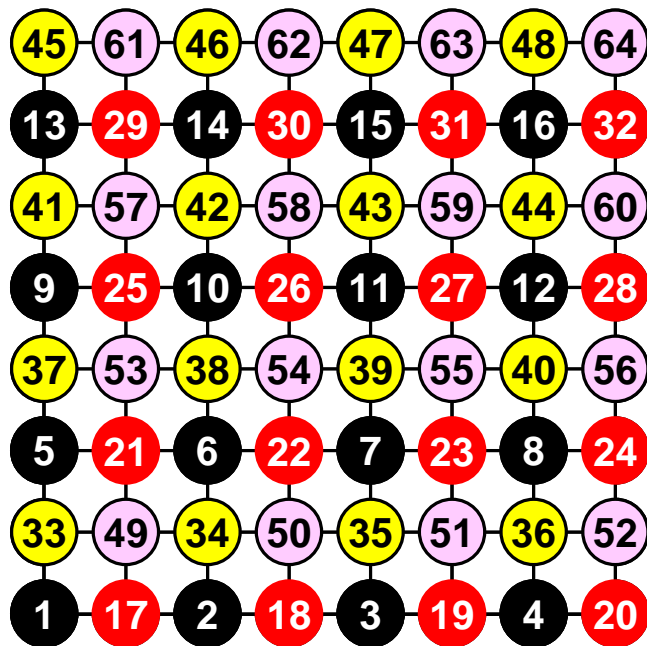


**CM-RCM (Color#=4)**  
Cyclic MC + RCM

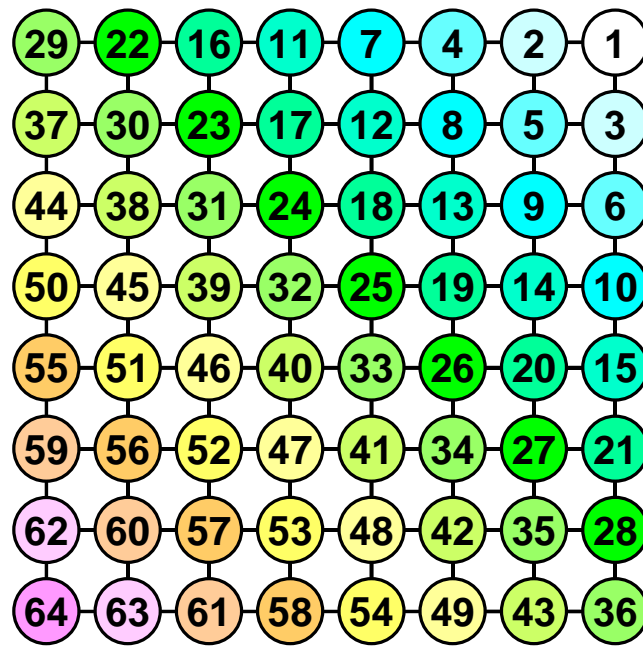


# Ordering Methods

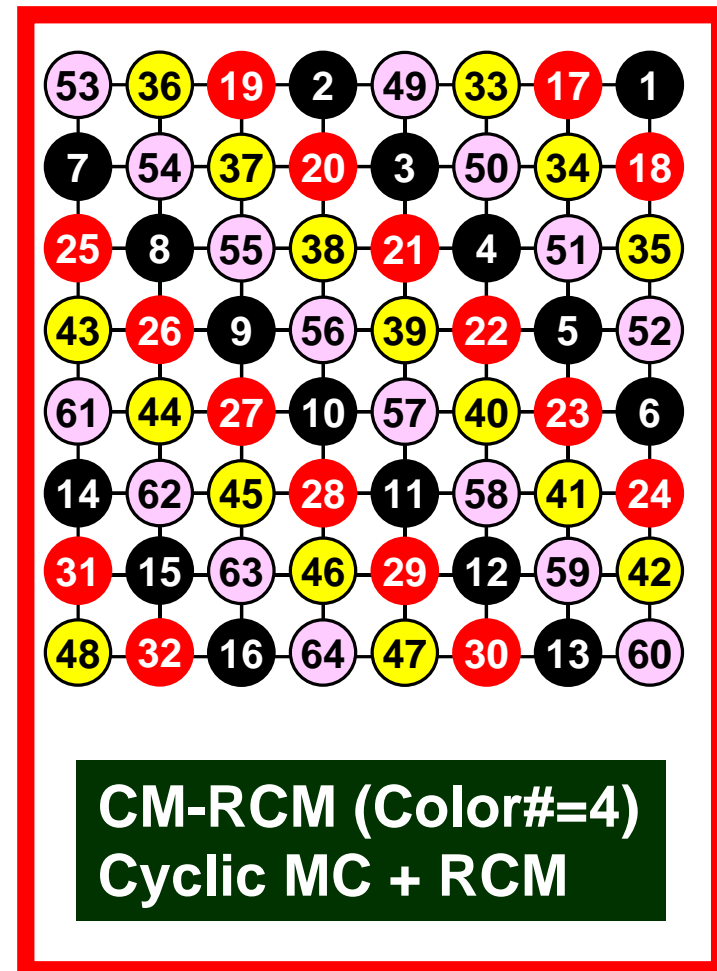
Elements in “same color” are independent: to be parallelized



**MC (Color#=4)  
Multicoloring**



**RCM  
Reverse Cuthill-Mckee**



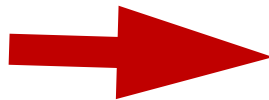
**CM-RCM (Color#=4)  
Cyclic MC + RCM**

# What is new in this work ?

- Storage format of coefficient matrices
  - CRS (Compressed Row Storage): Original
  - ELL (Ellpack-Itpack)
- Coarse Grid Aggregation (CGA)
- Hierarchical CGA: Communication Reducing CGA

# ELL: Fixed Loop-length, Nice for Pre-fetching

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



1	3	
1	2	5
4	1	3
3	7	4
1	5	

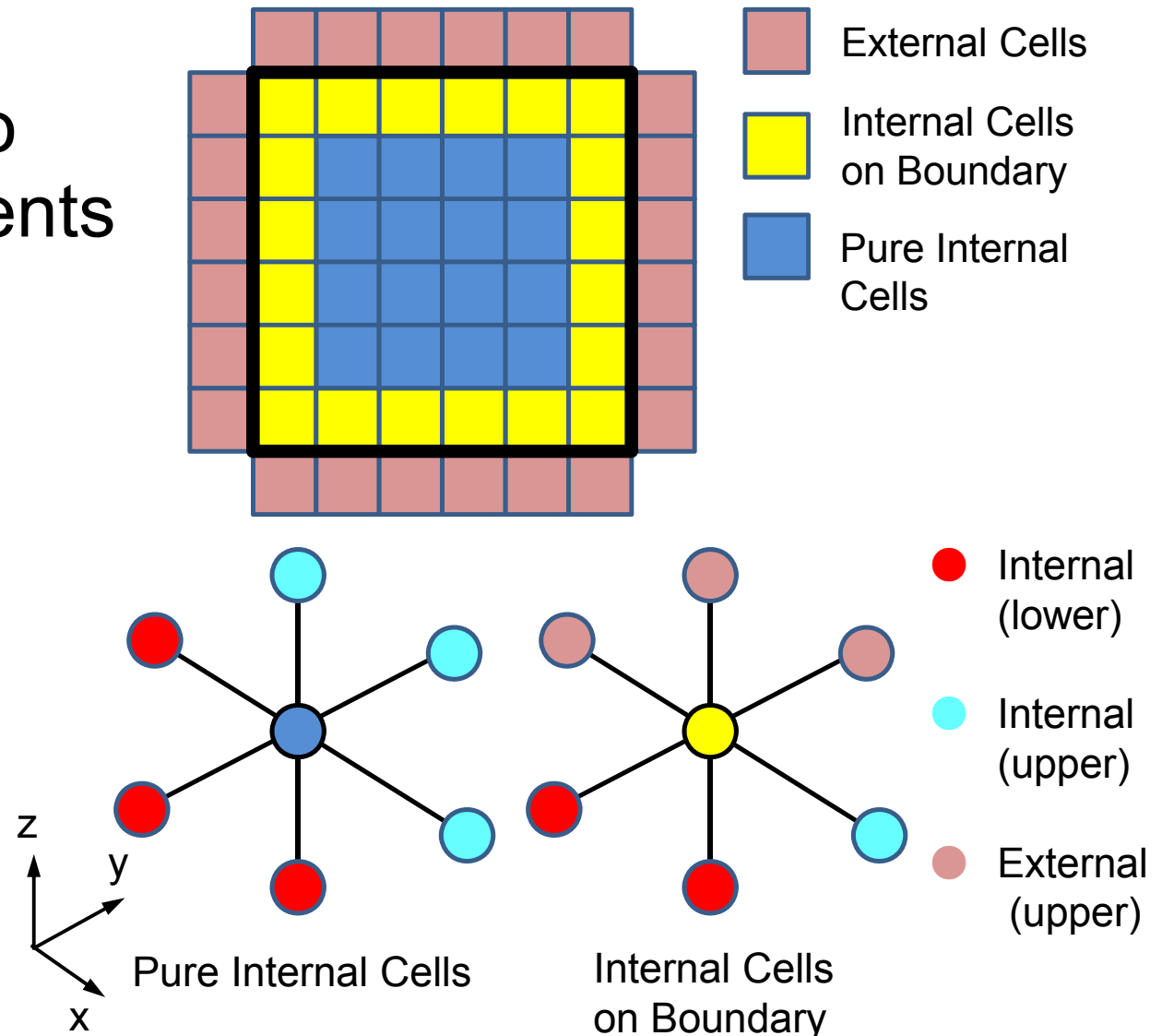
(a) CRS

1	3	0
1	2	5
4	1	3
3	7	4
1	5	0

(b) ELL

# Special Treatment for “Boundary” Cells connected to “Halo”

- Distribution of Lower/Upper Non-Zero Off-Diagonal Components
- Pure Internal Cells
  - L:  $\sim 3$ , U:  $\sim 3$
- Boundary Cells
  - L:  $\sim 3$ , U:  $\sim 6$



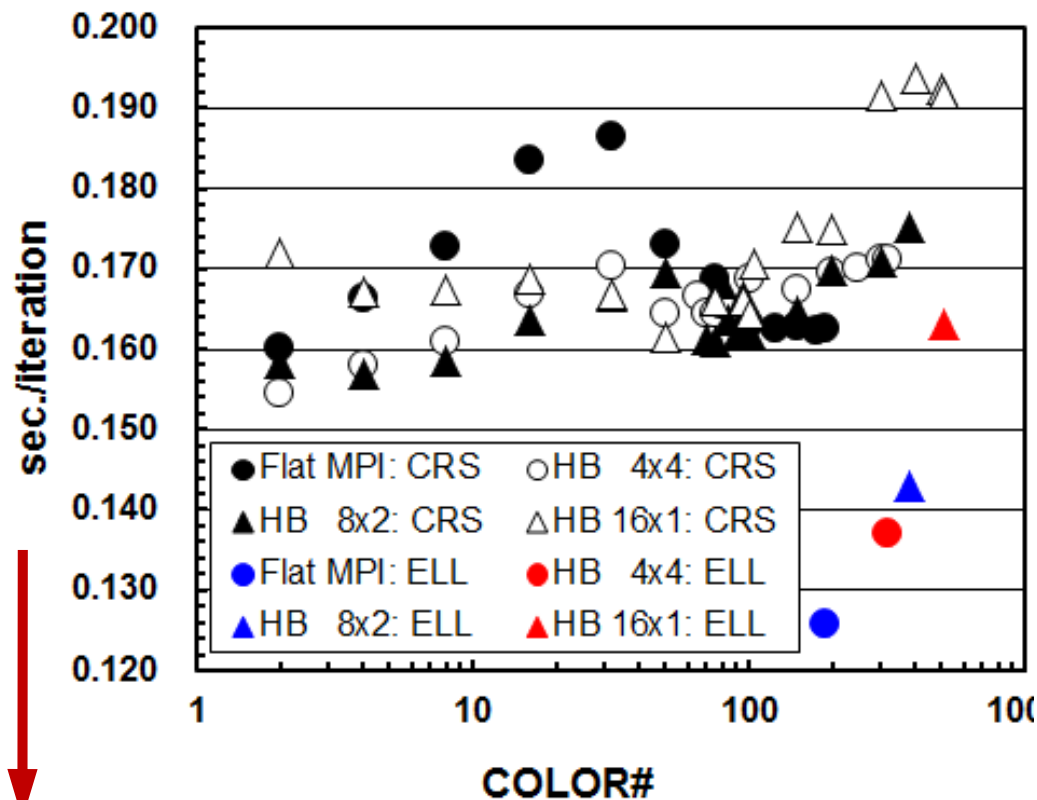
# Effect of CRS/ELL

4 nodes, 64 cores, (16,777,216 meshes:  $64^3$  meshes/core)

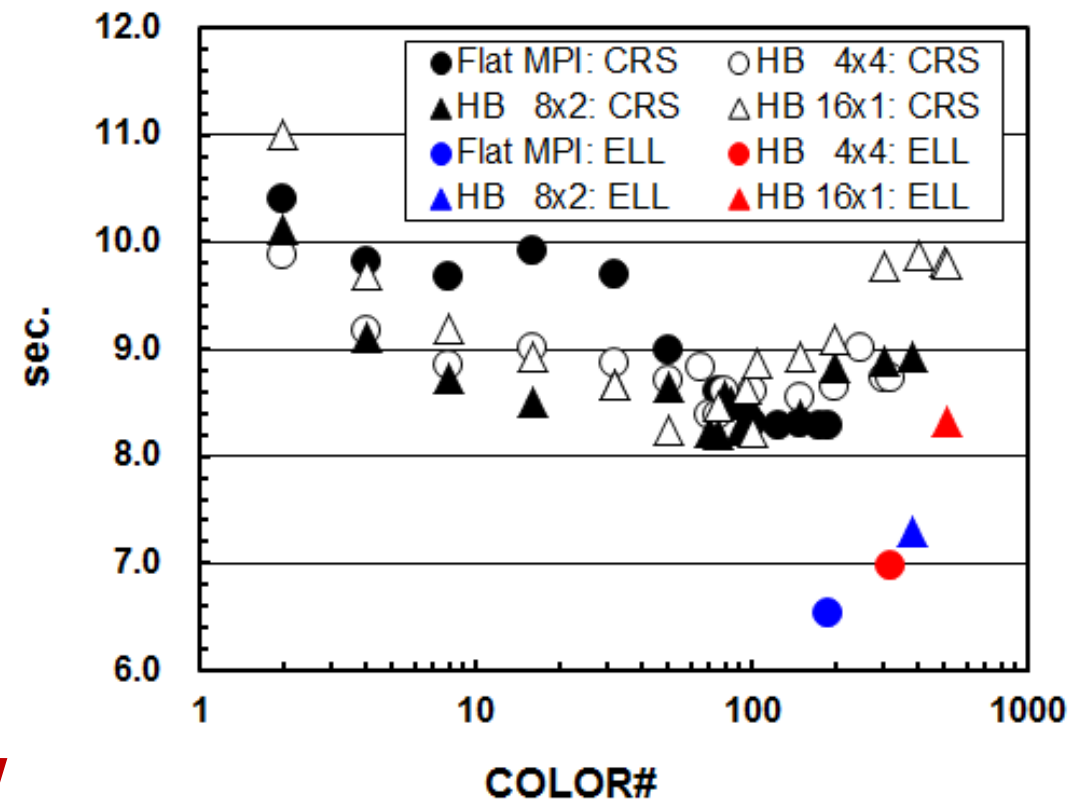
CM-RCM(k), only RCM for ELL cases

DOWN is GOOD

sec./iteration



time for MGCG



Down is good

# Analyses by Detailed Profiler of Fujitsu FX10, single node, Flat MPI, RCM (Multigrid Part)

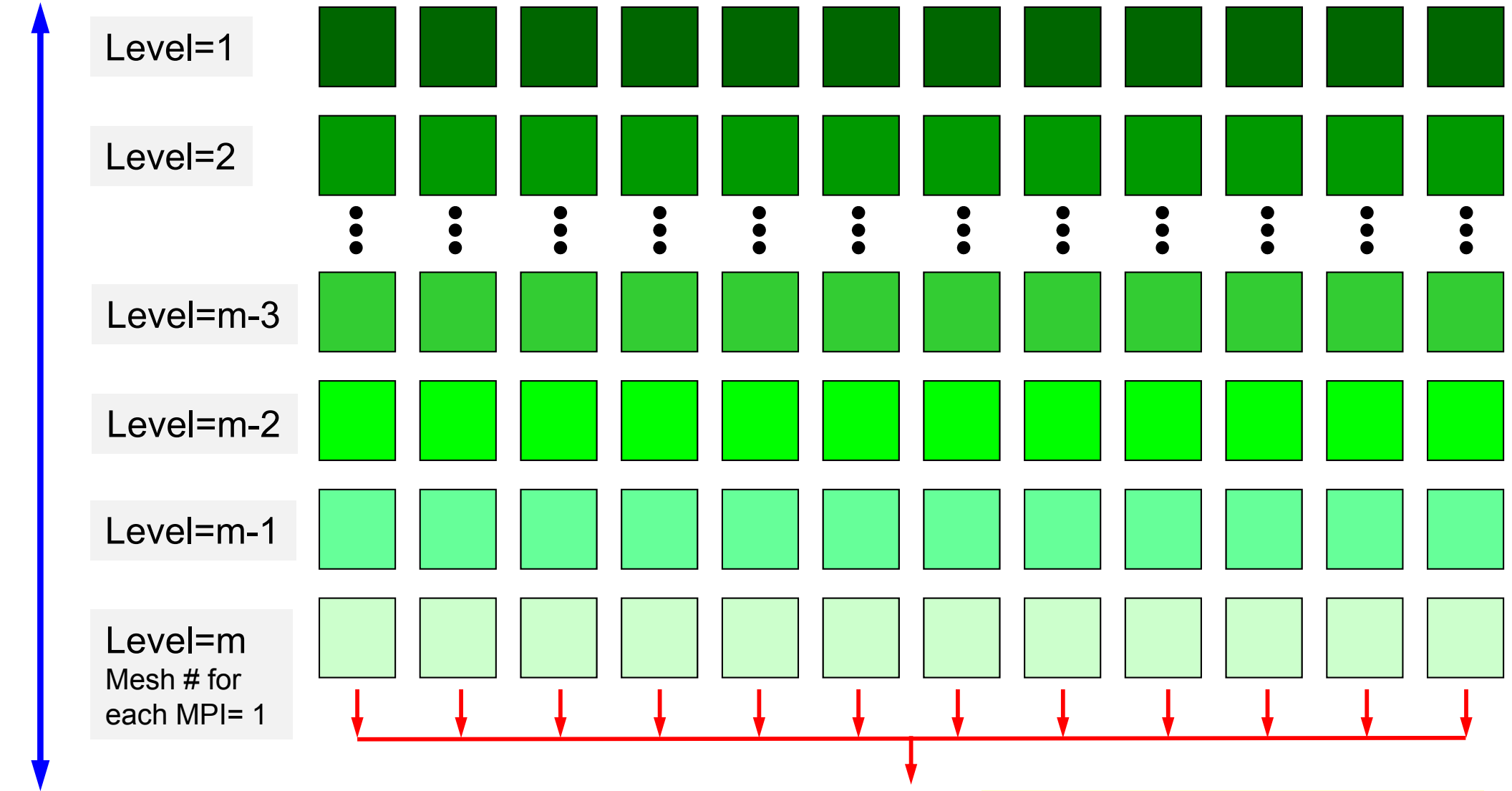
ELL with fixed loop length: accel. prefetching

	L1-cache Demand Miss	Instructions	Time for Multigrid	Operation Wait
CRS	29.3%	$1.447 \times 10^{10}$	6.815 sec.	1.453 sec.
ELL	16.5%	$6.385 \times 10^9$	5.457 sec.	0.312 sec.

# Original Approach (restriction)

Coarse grid solver at a single core [KN 2010]

Fine

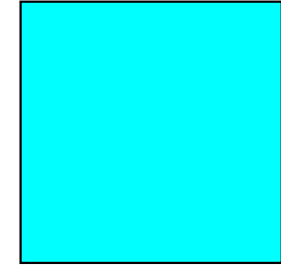
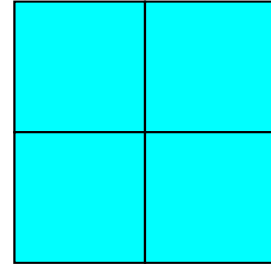
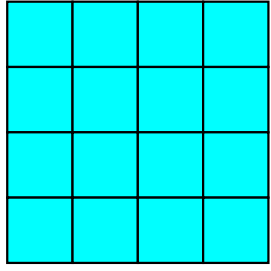
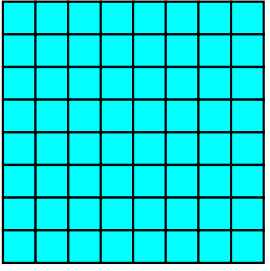


Coarse

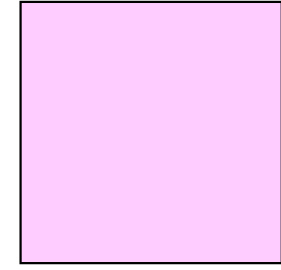
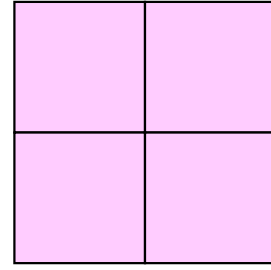
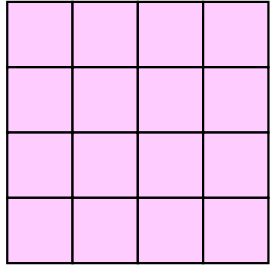
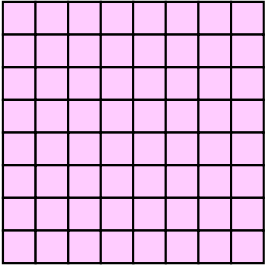
Coarse grid solver on a single core (further multigrid)

# Coarse Grid Solver on a Single Core

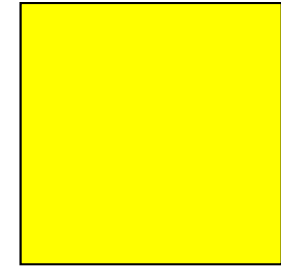
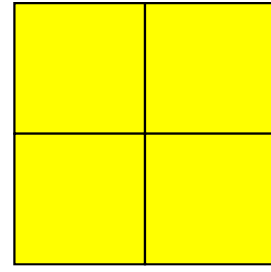
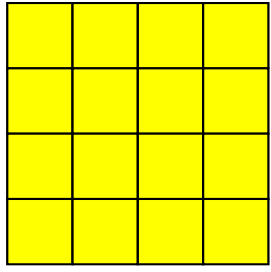
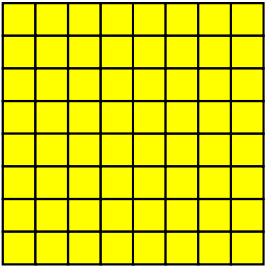
PE#0



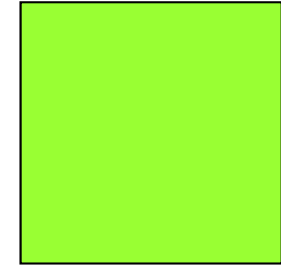
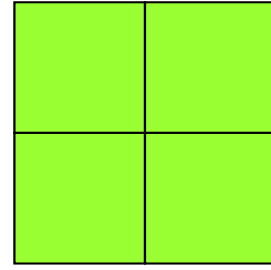
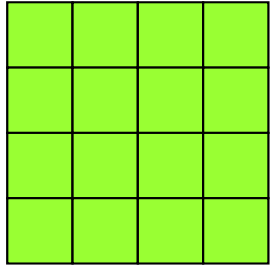
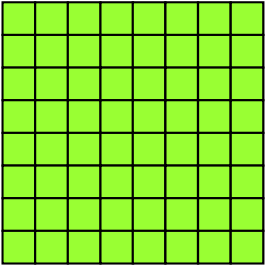
PE#1



PE#2



PE#3



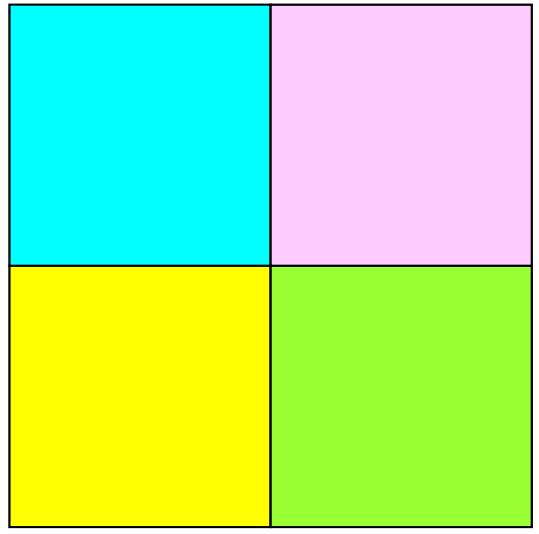
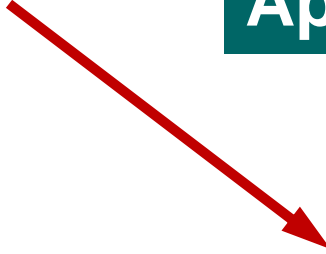
lev=1

lev=2

lev=3

lev=4

Original Approach



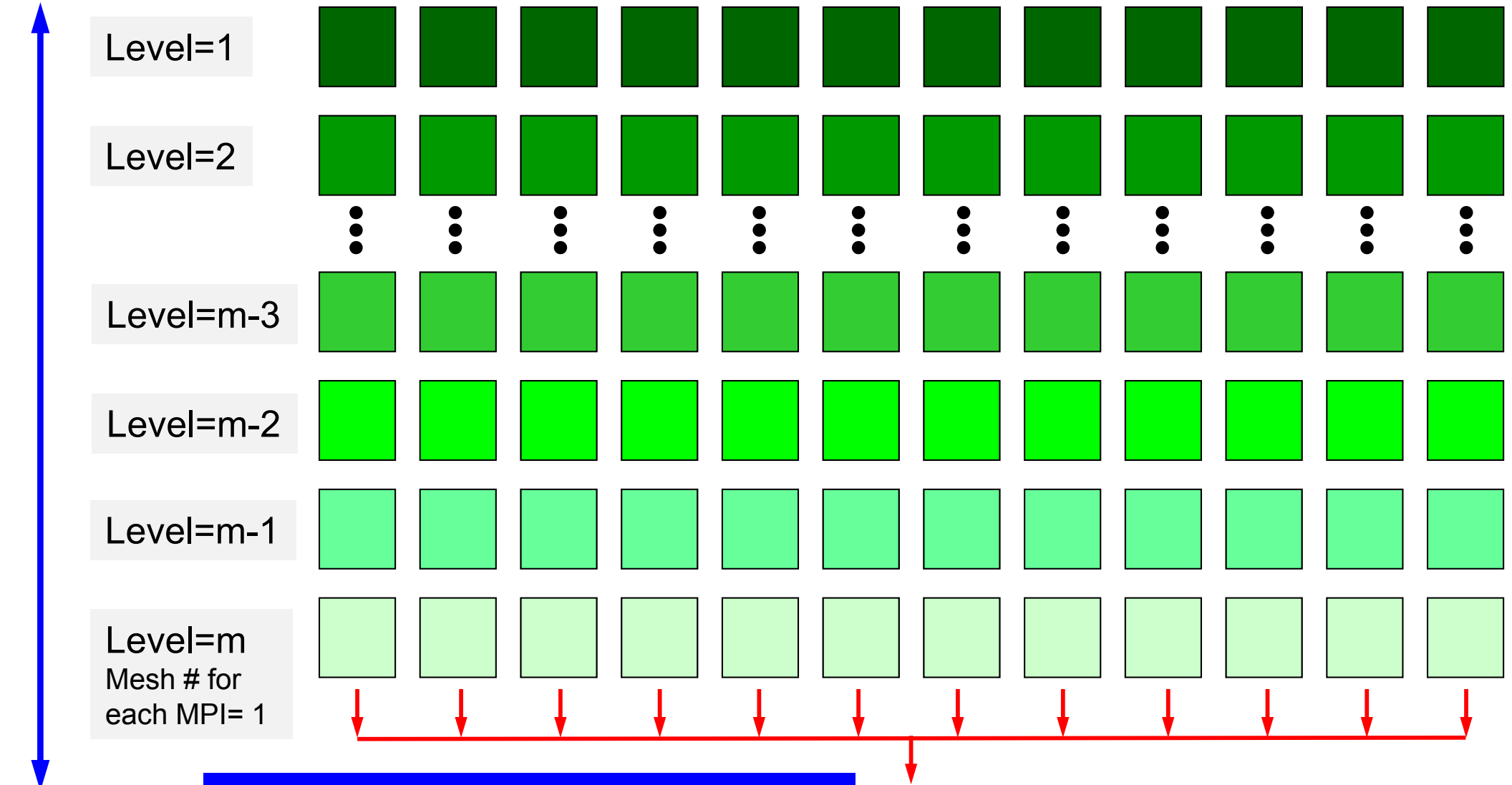
Size of the Coarsest Grid= Number of MPI Processes Redundant Process  
In Flat-MPI, this size is larger



# Original Approach (restriction)

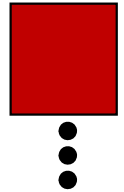
Coarse grid solver at a single core [KN 2010]

Fine



Coarse

**Communication Overhead at Coarser Levels**

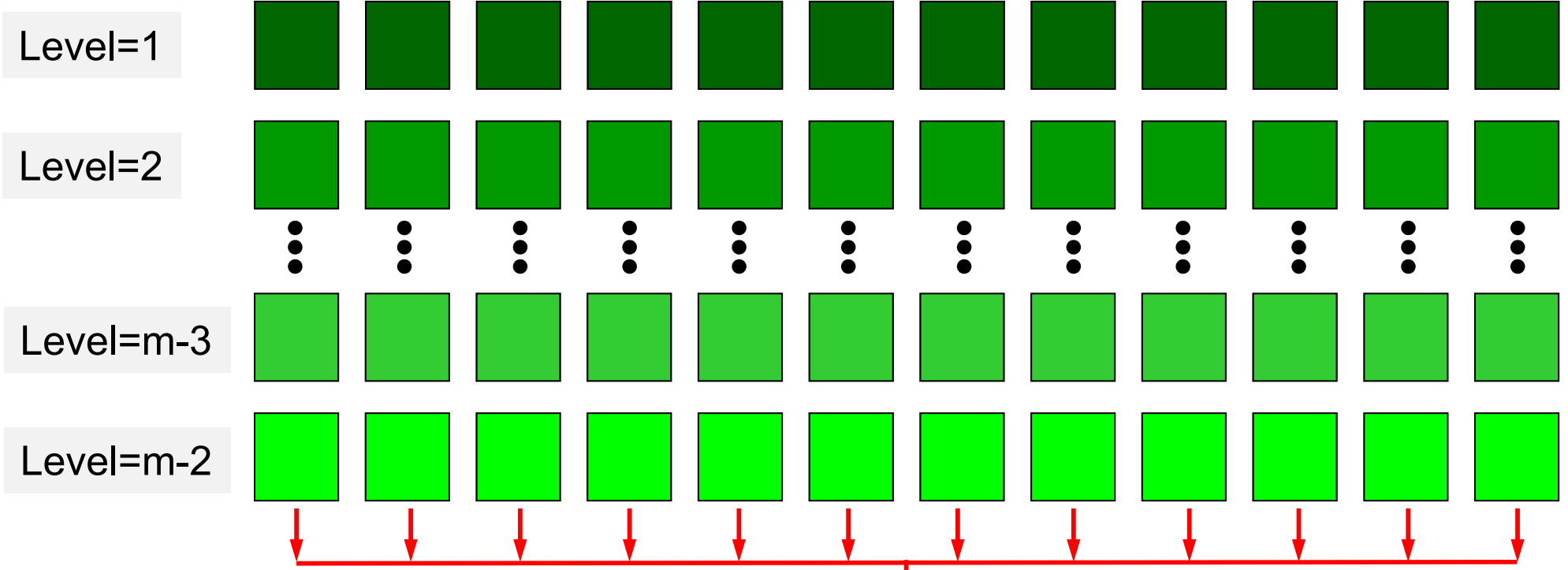


Coarse grid solver on a single core (further multigrid)

# Coarse Grid Aggregation (CGA)

Coarse Grid Solver is multithreaded [KN 2012]

Fine



- Communication overhead could be reduced
- Coarse grid solver is more expensive than original approach.
- If process number is larger, this effect might be significant

Coarse grid solver on a single MPI process (multi-threaded, further multigrid)

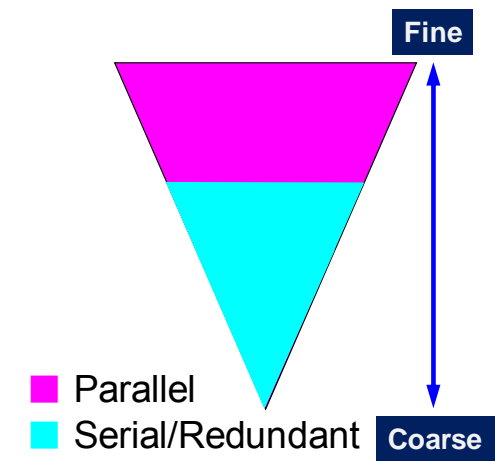
Coarse

# Results at 4,096 nodes

*lev.*: switching level to “coarse grid solver”

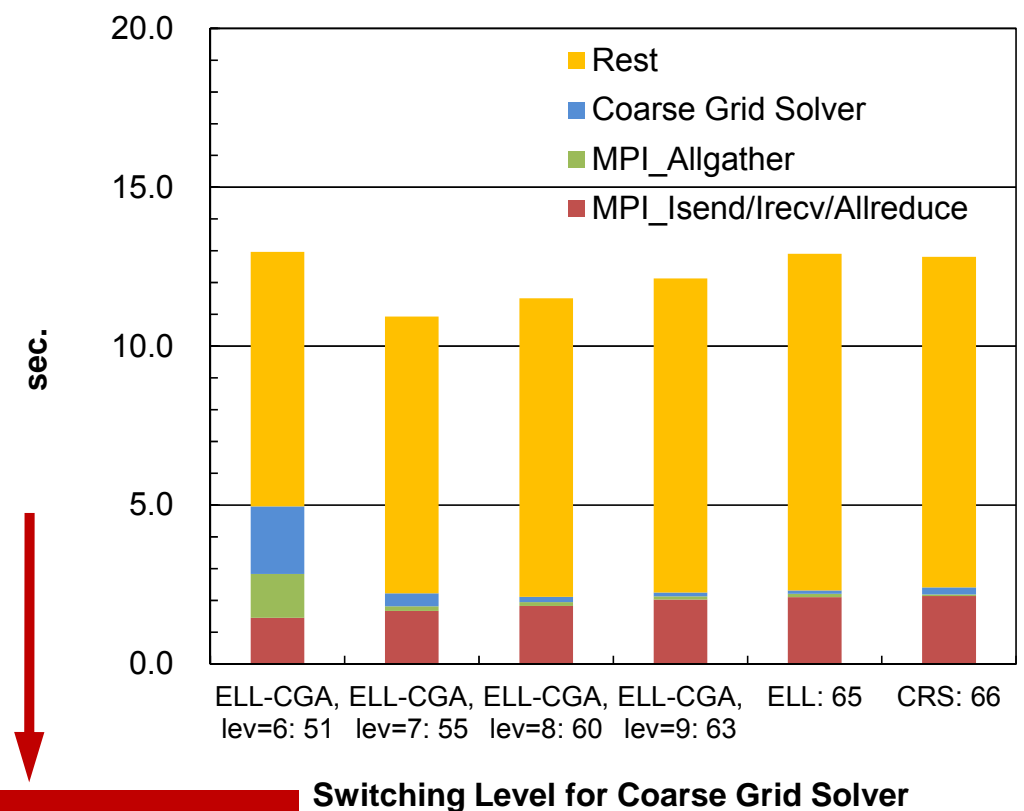
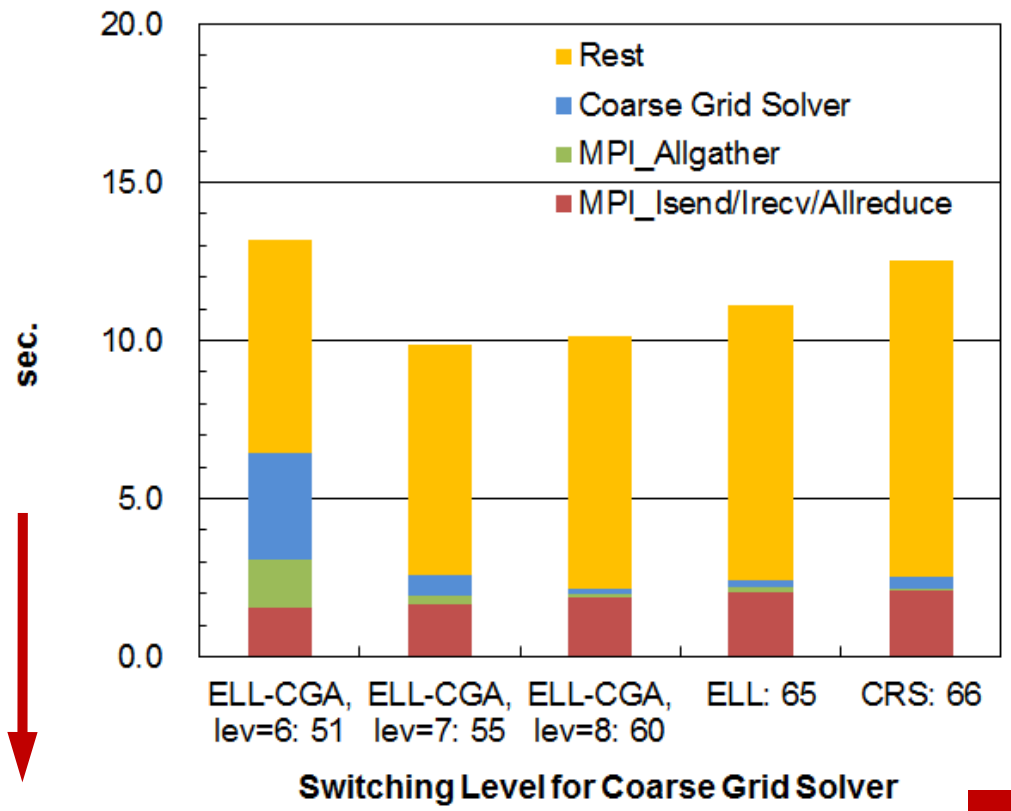
Opt. Level= 7, HB 8x8 is the best

**DOWN is GOOD**



## HB 8x2

## HB 16x1

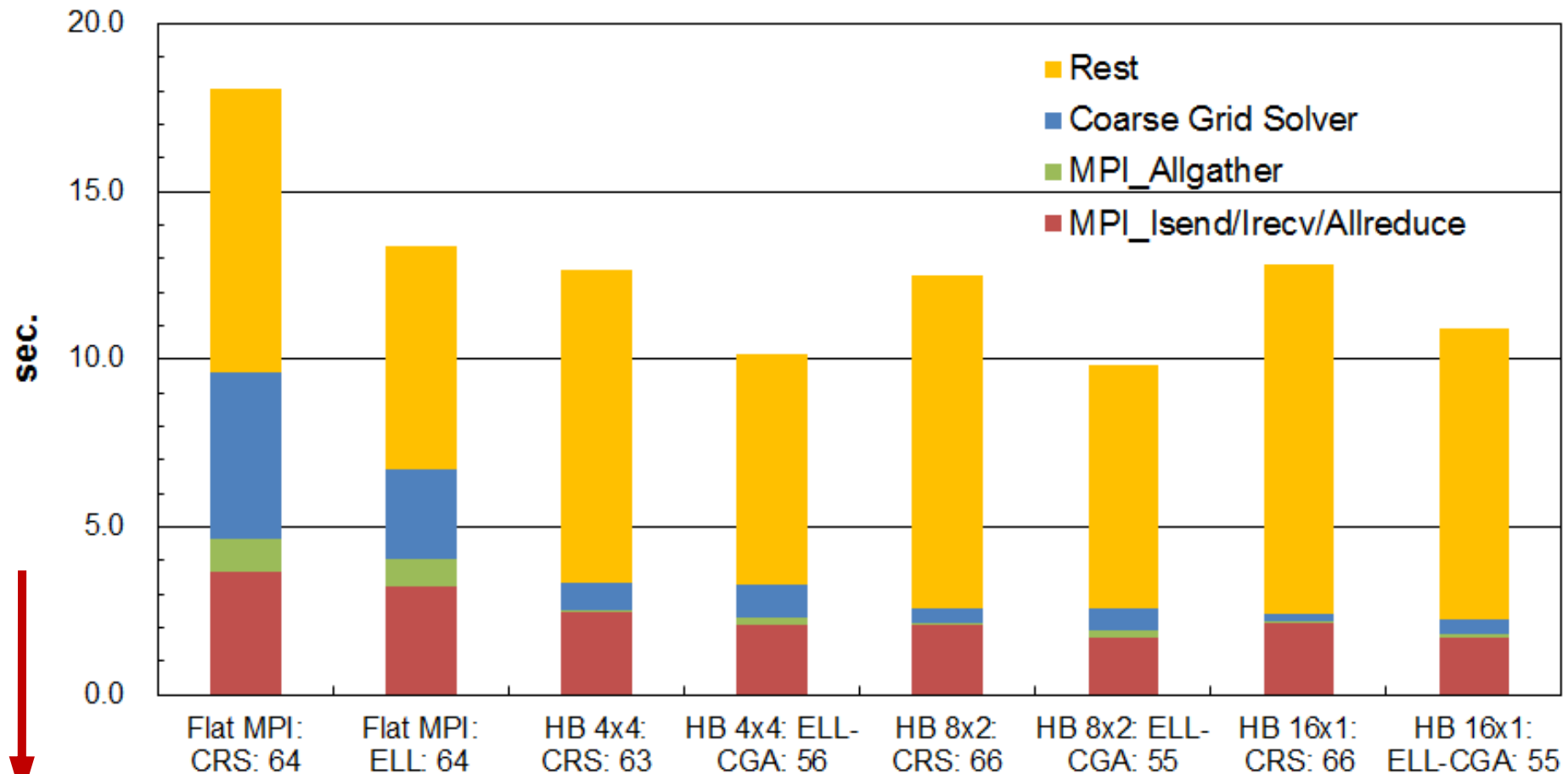


**Down is good**

# Weak Scaling at 4,096 nodes

17,179,869,184 meshes ( $64^3$  meshes/core)

best switching level (=7)



Down is good

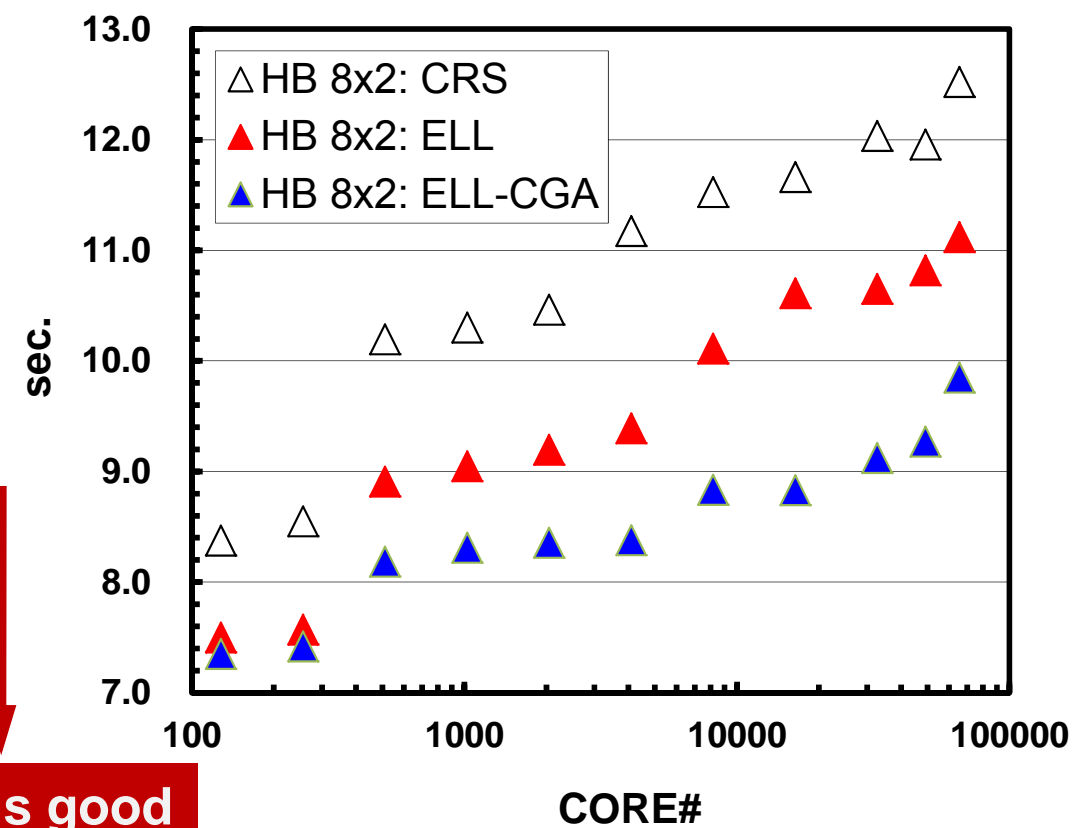
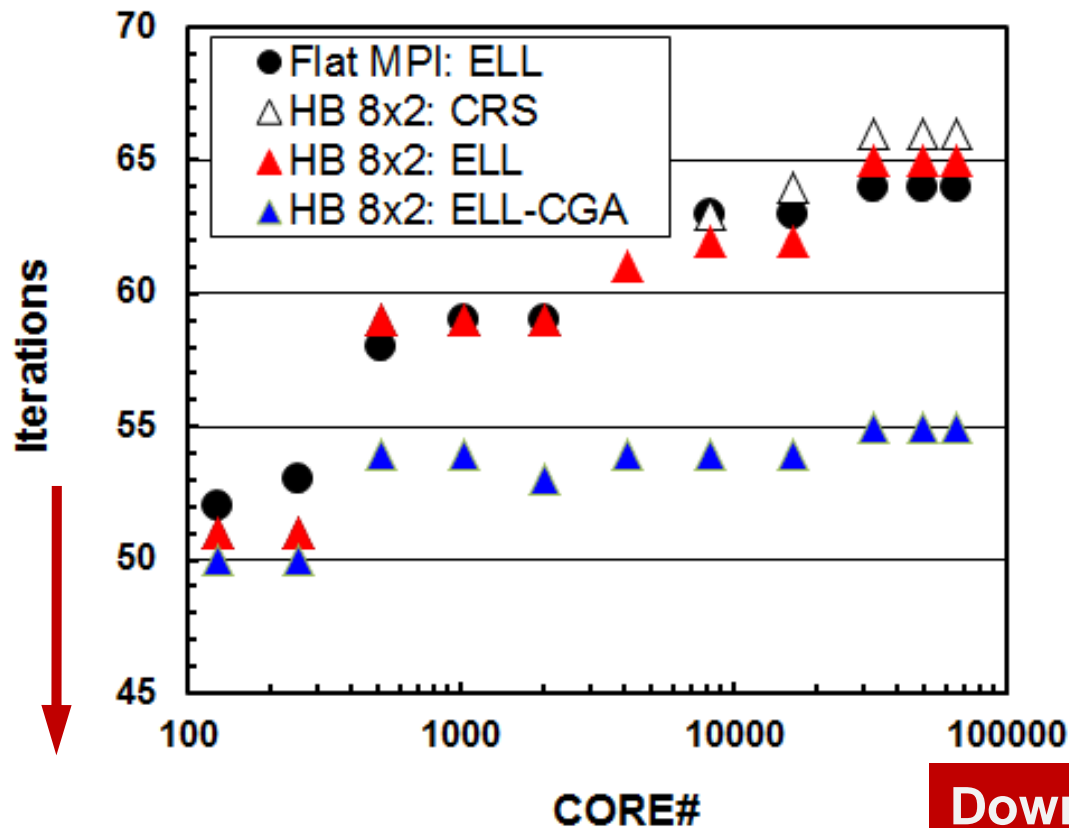
# Weak Scaling: up to 4,096 nodes

up to 17,179,869,184 meshes ( $64^3$  meshes/core)

Convergence has been much improved by coarse grid aggregation, DOWN is GOOD

Iterations

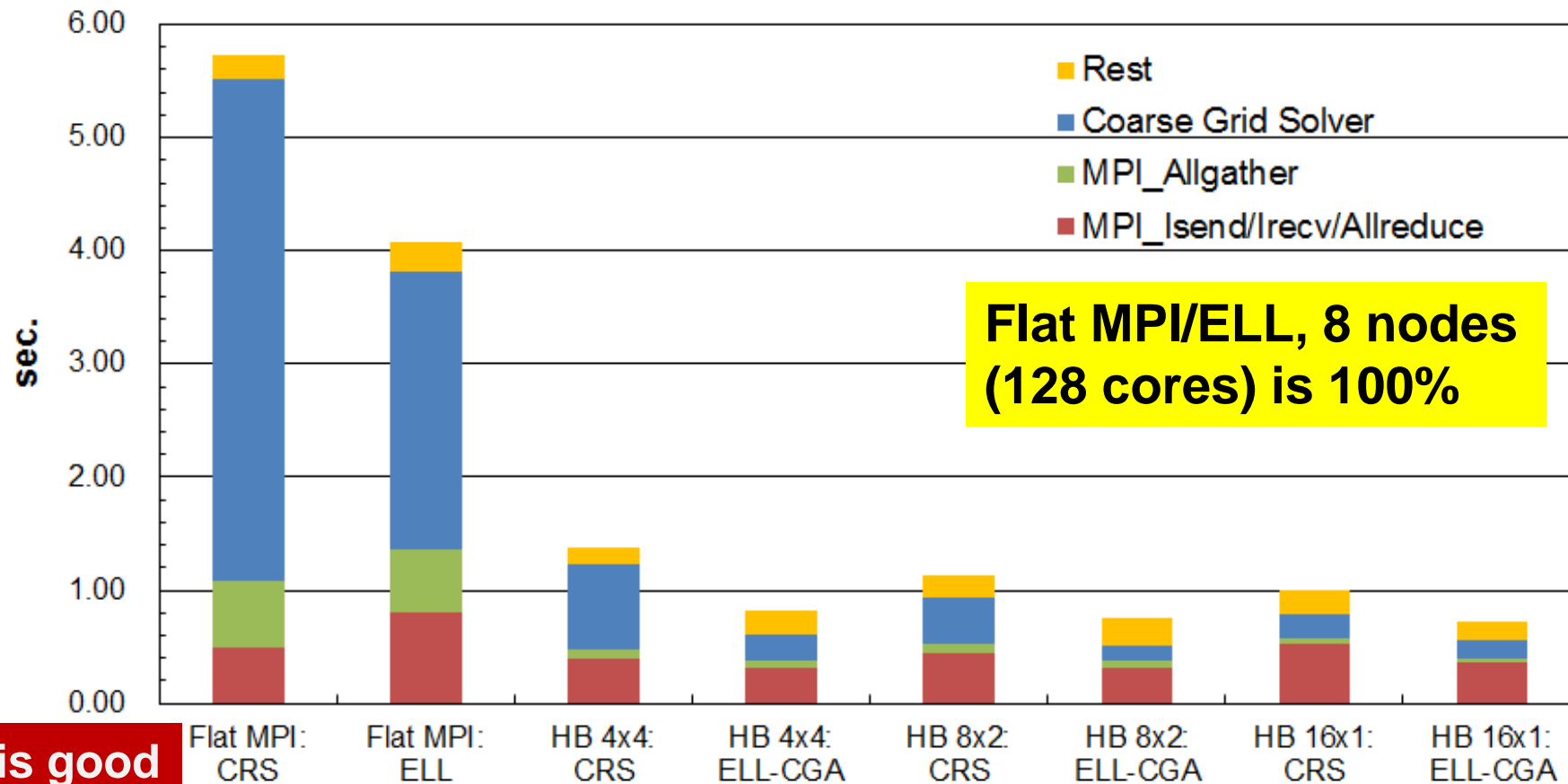
sec.



Down is good

# Strong Scaling at 4,096 nodes

268,435,456 meshes, only  $16^3$  meshes/core at 4,096 nodes

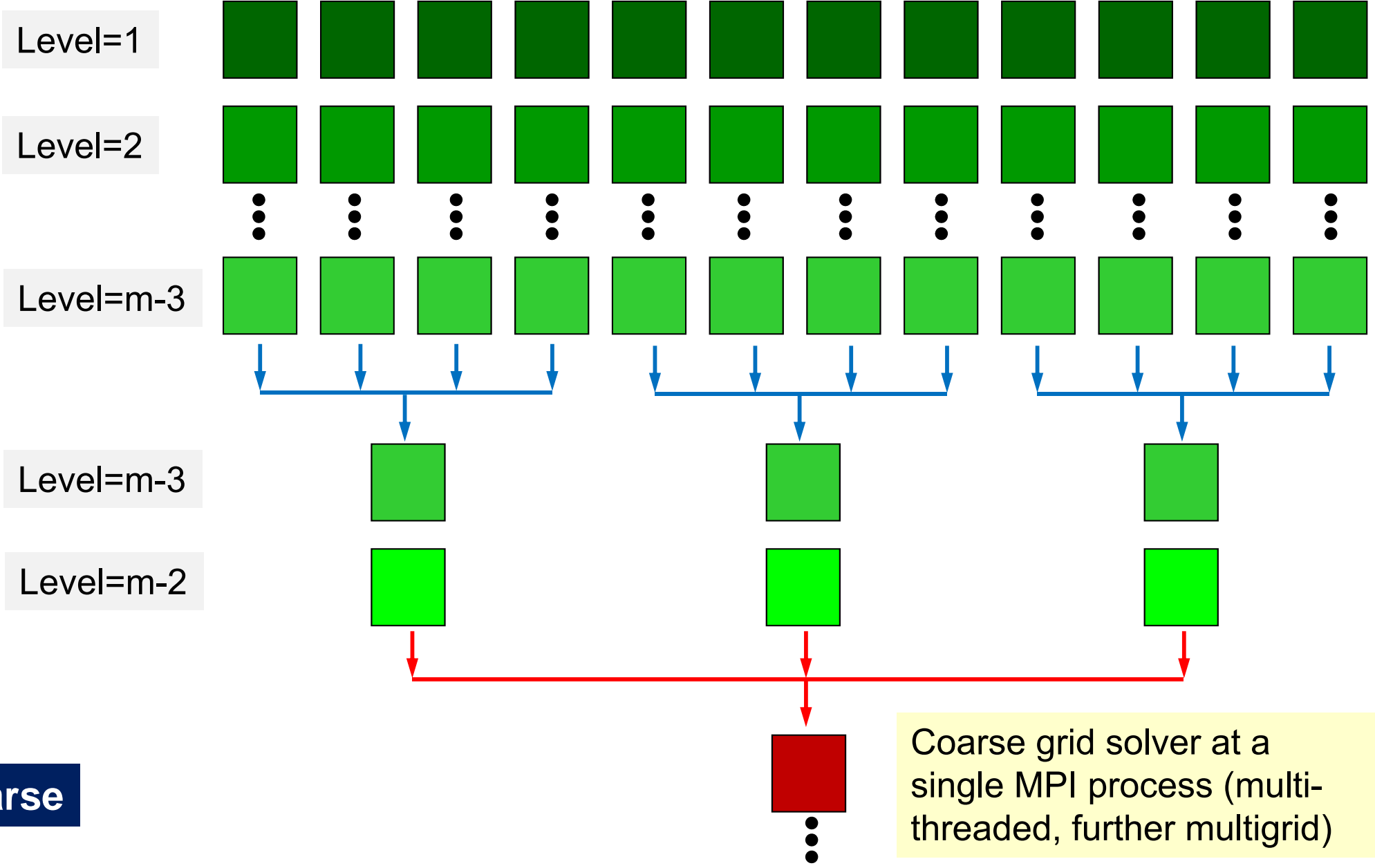


	Flat MPI		HB 4×4		HB 8×2		HB 16×1	
	CRS	ELL	CRS	ELL-CGA	CRS	ELL-CGA	CRS	ELL-CGA
Iterations until Convergence	57	58	58	46	63	49	63	51
MGCG solver (sec.)	5.73	4.07	1.38	.816	1.13	.749	1.00	.714
Parallel performance (%)	2.02	2.85	8.38	14.2	10.3	15.5	11.6	16.2

# Hierarchical CGA: Comm. Reducing MG

Fine

Reduced number of MPI processes[KN 2013]



Coarse

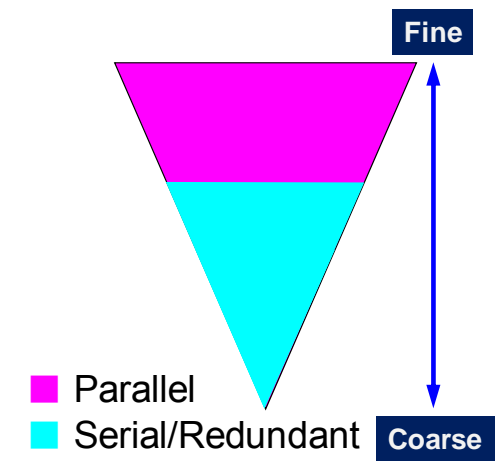
Coarse grid solver at a single MPI process (multi-threaded, further multigrid)

# Results at 4,096 nodes

*lev.*: switching level to “coarse grid solver”

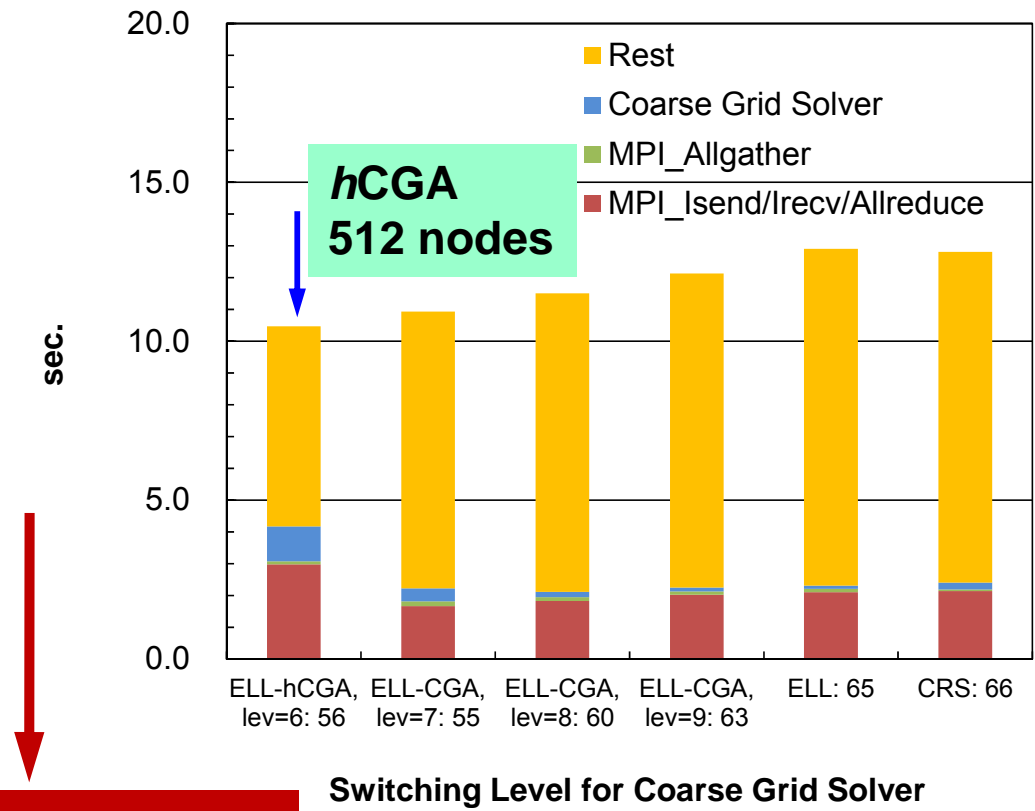
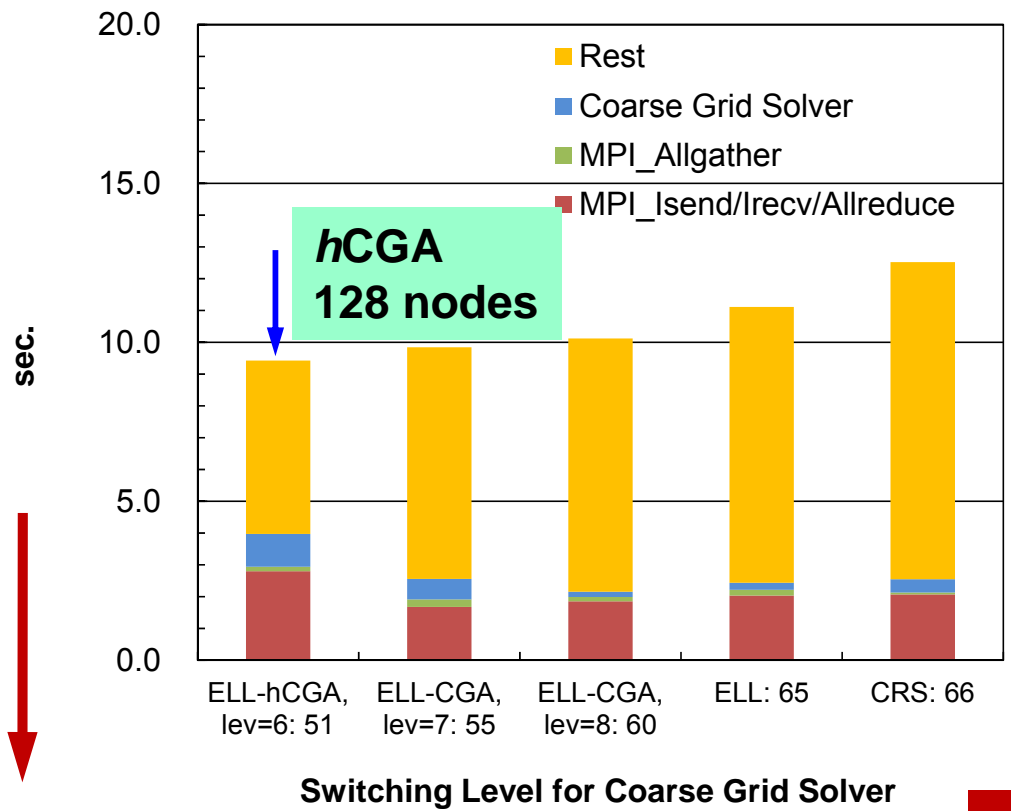
Opt. Level= 7, HB 8x8 is the best

**DOWN is GOOD**



## HB 8x2

## HB 16x1



**Down is good**

Switching Level for Coarse Grid Solver

Switching Level for Coarse Grid Solver



# Summary

- ELL format is effective !
- “Coarse Grid Aggregation (CGA)” is effective for stabilization of convergence at  $O(10^4)$  cores for MGCG
  - HB 8x2 is the best at 4,096 nodes
- Hierarchical CGA (*hCGA*) is also effective at 4,096 nodes
- Future/On-Going Works and Open Problems
  - Algorithms
    - CA-Multigrid (for coarser levels), CA-SPAI
  - Strategy for Automatic Selection
    - optimum switching level, number of processes for *hCGA*, optimum color #
  - More Flexible ELL for Unstructured Grids
  - Optimized MPI (co-design)
    - e.g. MPI on Fujitsu FX10 utilizing RDMA with persistent communications
  - Optimum number of colors
    - strongly depends on thread #, H/W etc ...