東京大学
THE UNIVERSITY OF TOKYO

東京大学情報基盤センター
INFORMATION TECHNOLOGY CENTER, THE UNIVERSITY OF TOKYO

Tokyo Woman's Christian University
東京女子大学

Hierarchical, Hybrid, Heterogeneous
h3-Open-BDEC
Big Data & Extreme Computing

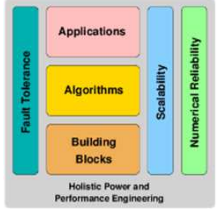
RIKEN

R-CCS

Efficient Parallel Multigrid Methods on Manycore Clusters with Double/Single Precision Computing

Kengo Nakajima[*1,*2], Takeshi Ogita[*3], Masatoshi Kawai[*1]

*1: Information Technology Center, University of Tokyo, *2: RIKEN R-CCS,
*3: Tokyo Women's Christian University



iWAPT 2021 in conjunction with IPDPS 2021
May 21, 2021 (online)

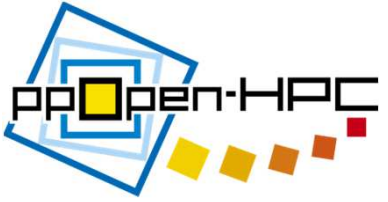
ppOpen-HPC

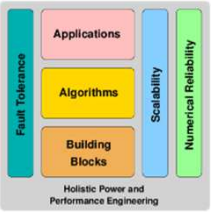
Acknowledgements

- JST/CREST
- DFG/SPPEXA
- JSPS Grant-in-Aid for Scientific Research (S): 19H05662
- JHPCN (jh200036-MDHI, jh200037-NAH, jh200041-NAH)
- JCAHPC

独立行政法人
科学技術振興機構
Japan Science and Technology Agency

Deutsche
Forschungsgemeinschaft

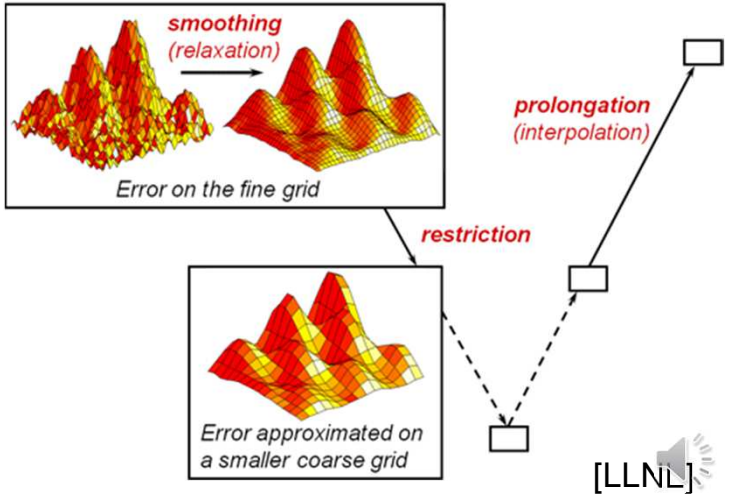




- Background & Overview
  - Multigtrid Methods
  - Overview of Previous/Present Works
- SELL-C-σ with Double Precision Computing
- Computing in Double/Single Precision
- Accuracy Verification
- Summary

Features of Multigrid (MG) Methods

- ✓ **Scalable multilevel method for solving linear equations**
- ✓ GMG (Geometrical Multigrid) and AMG (Algebraic Multigrid)
- ✓ **Number of iterations until convergence for multigrid method is kept constant as the problem size changes, Comp. Time = O(N)**
- ✓ The parallel multigrid method is expected to be one of the most powerful tools on exa-scale systems.
- ✓ Applied to rather well-conditioned problems (e.g. Poisson's eqn's)
  - ☐ Many sophisticated methods for real-world applications are under development (next presentation)
- ✓ **MG is scalable, but there are many things to be done towards exascale computing**

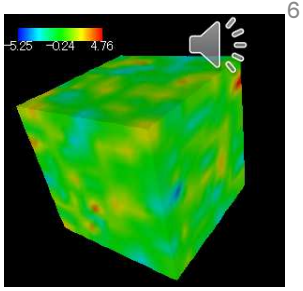


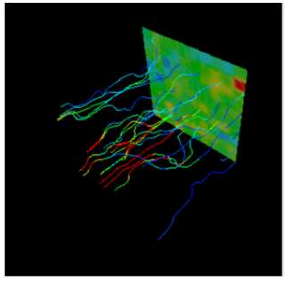
[LLNL]

Overview (1/3)

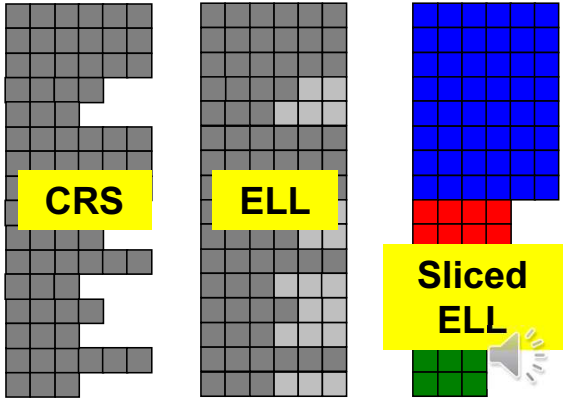
- The parallel multigrid method is expected to play an important role in scientific computing on exa-scale supercomputer systems for solving large-scale linear equations with sparse coefficient matrices.
- Because solving sparse linear systems is a very memory-bound process, efficient method for storage of coefficient matrices is a crucial issue.
- In the previous works, authors implemented **Sliced ELL** method to parallel conjugate gradient solvers with multigrid preconditioning (**MGCG**) for the application on 3D groundwater flow through heterogeneous porous media (pGW3D-FVM), and excellent performance has been obtained on large-scale multicore/manycore clusters.

pGW3D-FVM: Target Application
[KN IEEE ICPADS 2014] (Best Paper Award)





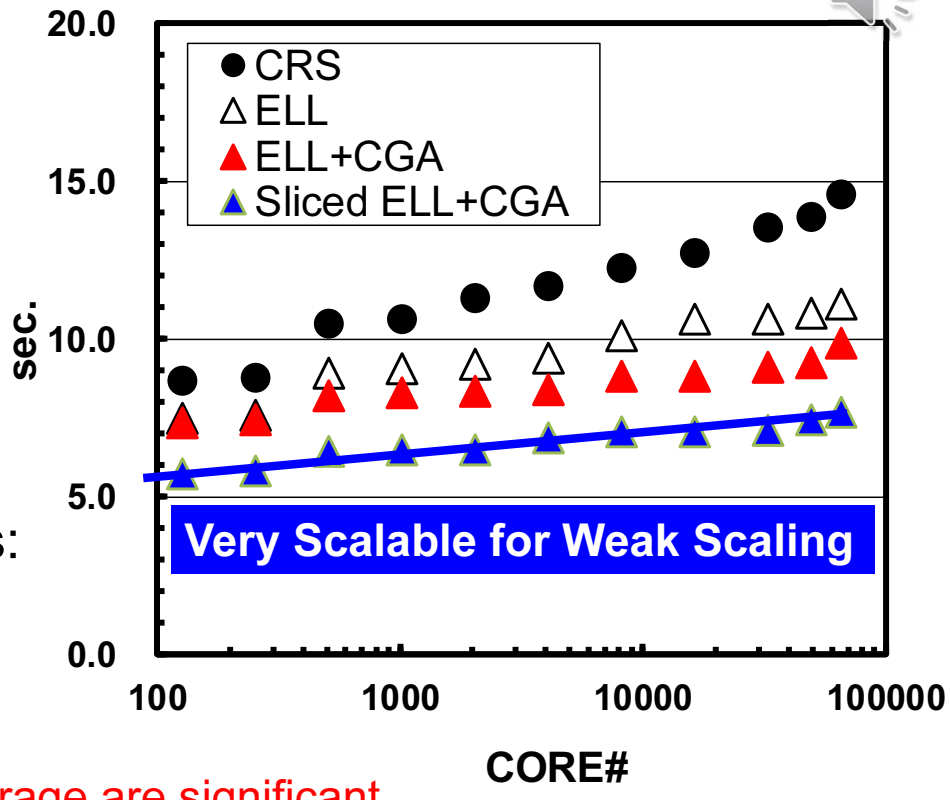
- 3D Groundwater Flow via Heterogeneous Porous Media
- Finite-Volume Method on Structured Cubic Voxel Mesh
- Poisson's equation $\nabla \cdot (\lambda(x,y,z)\nabla\phi) = q$
  - Randomly distributed water conductivity (λ)
  - $\lambda=10^{-5}\sim10^{+5}$, Average: 1.00
- **Conjugate Gradient preconditioned by Multigrid (MGCG)**
  - **Geometric Multigrid (GMG): Octree-based**
  - **IC(0) Smoother, V-Cycle**
  - **Additive Schwartz Domain Decomposition**
- **Sliced ELL for Storage of Sparse Matrices**
- **Fortran90, MPI/OpenMP**



Nakajima, K., Optimization of Serial and Parallel Communications for Parallel Geometric Multigrid Method, Proceedings of the 20th IEEE International Conference for Parallel and Distributed Systems (ICPADS 2014) 25-32, Hsin-Chu, Taiwan, 2014

Effects of Sliced-ELL on MGCG/pGW3D-FVM [KN ICPADS 2014]

- Fujitsu PRIMEHPC FX10
  - Weak-Scalingup to 4,096-nodes, 655,536-cores
  - max 17,179,869,184 DOF
  - HB 8x2
- 1.9x performance@655,536-cores: (Sliced ELL+CGA) over CRS
- MGCG: Sparse Linear Solver
  - Typical Memory-Bound Procedure
  - Effects of Memory Access/Matrix Storage are significant



Nakajima, K., Optimization of Serial and Parallel Communications for Parallel Geometric Multigrid Method, Proceedings of the 20th IEEE International Conference for Parallel and Distributed Systems (ICPADS 2014) 25-32, Hsin-Chu, Taiwan, 2014

Overview (2/3)

- In the present work, authors introduced SELL-C-σ with double/single precision computing to the MGCG solver, and evaluated the performance of the solver with OpenMP/MPI hybrid parallel programing models on the Oakforest-PACS (OFP) system at JCAHPC **using up to 2,048 nodes of Intel Xeon Phi**.
- **Because SELL-C-σ is suitable for wide-SIMD architecture, such as Xeon Phi, improvement of the performance over the sliced ELL was more than 35% for double precision and more than 45% for single precision on OFP.**
- **Finally, accuracy verification was conducted based on the method proposed by authors for solving linear equations with sparse coefficient matrices with M-property.**

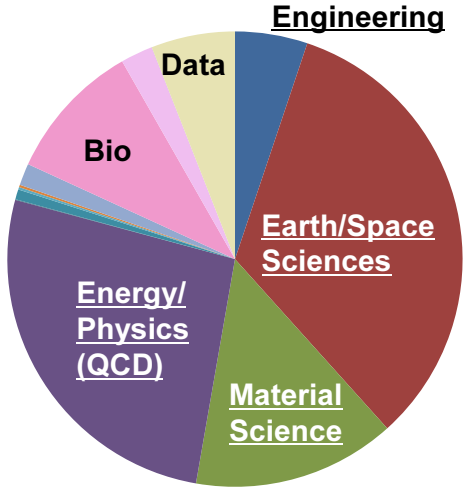
Overview (3/3)

- This is one of the first examples of SELL-C-σ applied to forward/backward substitutions in ILU-type smoother of multigrid solver with double/single precision computing.
- The effect of SELL-C-σ for computing with single precision (FP32) is very significant.

- Background & Overview
  - Multigtrid Methods
  - Overview of Previous/Present Works
- **SELL-C-σ with Double Precision Computing**
- Computing in Double/Single Precision
- Accuracy Verification
- Summary

Target System: Oakforest-PACS (OFP)

- **Intel Xeon Phi (Knights Landing, KNL), OPA**, Fujitsu
- 8,208 nodes, 25+PF, 22nd in TOP 500 (November 2020)
- Operated by JCAHPC (U.Tsukuba & U.Tokyo)



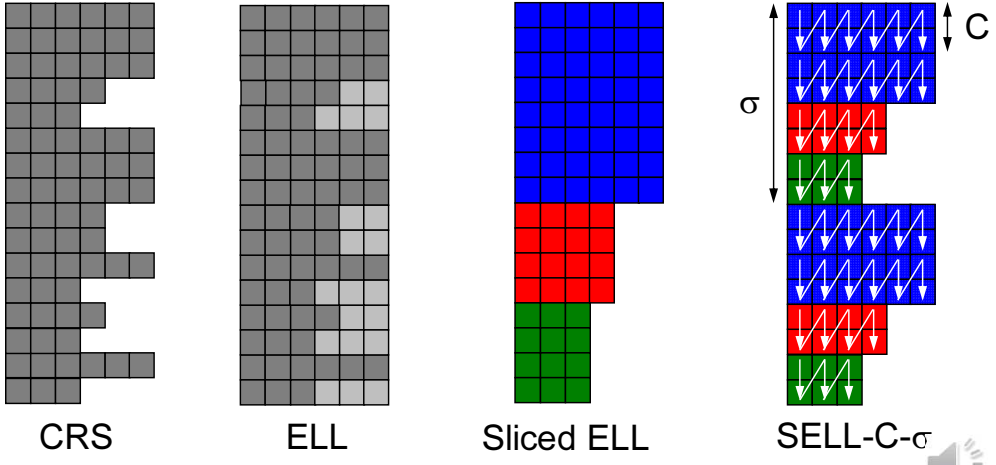
Usage in FY.2020

- Engineering
- Earth/Space
- Material
- Energy/Physics
- Info. Sci. : System
- Info. Sci. : Algrorithms
- Info. Sci. : AI
- Education
- Industry
- Bio
- Bioinformatics
- Social Sci. & Economics
- Data

JCAHPC 筑波大学 University of Tsukuba 東京大学 THE UNIVERSITY OF TOKYO

Main Purpose of the Present Work: SELL-C-σ

- **Applying SELL-C-σ to MGCG Solvers in pGW3D-FVM**
  - **Currently with Sliced ELL**
- Sliced ELL and SELL-C-σ have been mostly applied to SpMV, or Gauss-Seidel Iterative Solvers/Smoothers
- Implementation to Forward/Backward Substitution in ILU-type Smoothers is very difficult
  - First example of Sliced ELL [KN IEEE ICPADS 2014]
  - This is the first example of SELL-C-σ

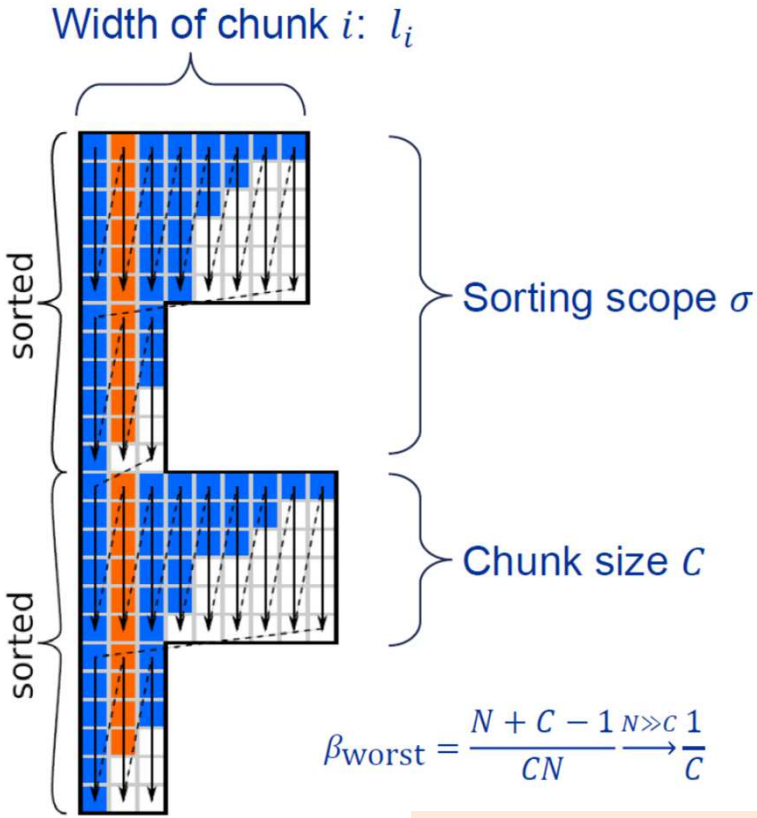


Constructing SELL-C-σ

1. Pick chunk size C (guided by SIMD/T widths)
2. Pick sorting scope σ
3. Sort rows by length within each sorting scope
4. Pad chunks with zeros to make them rectangular
5. Store matrix data in "chunk column major order"

"Chunk occupancy": fraction of "useful" matrix entries

$$\beta = \frac{N_{nz}}{\sum_{i=0}^{N_c} C \cdot l_i}$$

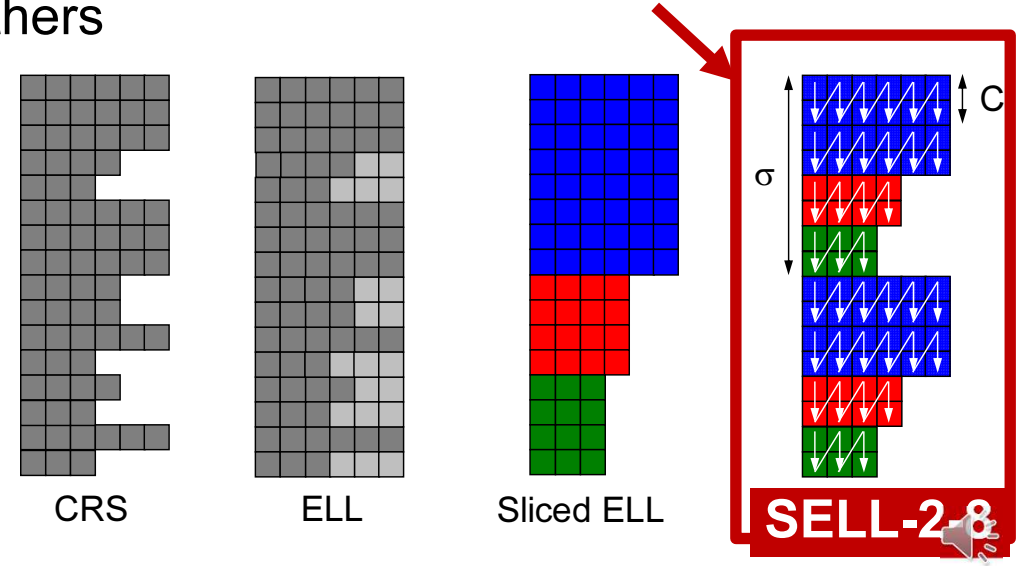


SELL-6-12
β=0.66

[Kreutzer, Hager, Wellein, SIAM SISC 2014]

Main Purpose of the Present Work: SELL-C-σ

- Applying SELL-C-σ to MGCG Solvers in pGW3D-FVM
  - Currently with Sliced ELL
- Sliced ELL and SELL-C-σ have been mostly applied to SpMV, or Gauss-Seidel Iterative Solvers/Smoothers
- **Implementation to Forward/Backward Substitution in ILU-type Smoothers is very difficult**
  - **First example of Sliced ELL [KN IEEE ICPADS 2014]**
  - **This is the first example of SELL-C-σ**



Coalesced，CM-RCM(2)

- CM-RCM with 2 colors: CM-RCM(2)
  - Number of iterations will increase with 2-colors, compared to RCM (current method)
  - Performance on OFP with multithreading is better with fewer colors
    - Loop length
  - Implementation of SELL-C-σ is easier than RCM
- Coloring part is not parallelized, but implementation of CM-RCM(2) is easy

Red-Black，2色のMC

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

CM-RCM(2)，Coalesced

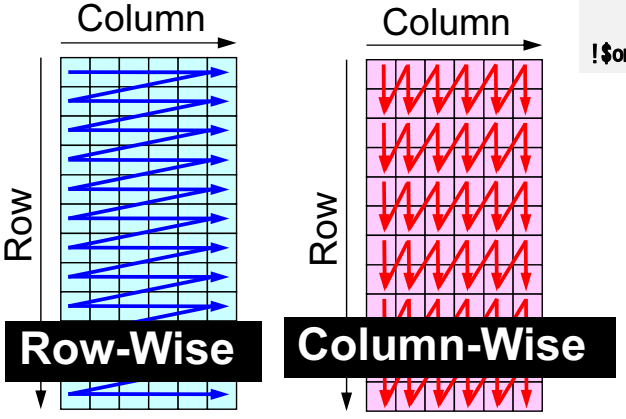
13	42	7	37	3	34	1	33
49	14	43	8	38	4	35	2
21	50	15	44	9	39	5	36
56	22	51	16	45	10	40	6
27	57	23	52	17	46	11	41
61	28	58	24	53	18	47	12
31	62	29	59	25	54	19	48
64	32	63	30	60	26	55	20

Forward Substitution: 2nd Color of CM-RCM(2)

Sliced-ELL Row-wise

(a) Sliced ELL: Row-Wise

```
!$omp parallel private (icol,…)
      icol= NCOLORtot
      (Operations)
!$omp do
      do (Loops:Meshes: Row-Wise)
      do (Loops:Non-Zero Off-Diag' s: j= 1, 6)
        (Operations)
      enddo
      enddo
!$omp end parallel
```



SCS-a (SELL-C-σ) Row-wise

(b) SELL-8-8/Row-Wise (SCS-a)

```
!$omp parallel private (icol,…)
      icol= NCOLORtot
      (Operations)
!$omp do
      do (Loops:Threads: ip= 1, PEsmpTOT)
      do (Loops:Blocks)
!$omp simd
        do (Loops SIMD: k= 1, 8)
        do (Loops:Non-Zero Off-Diag' s: j= 1, 6)
          (Operations)
        enddo
        enddo
      enddo
      enddo
!$omp end parallel
```

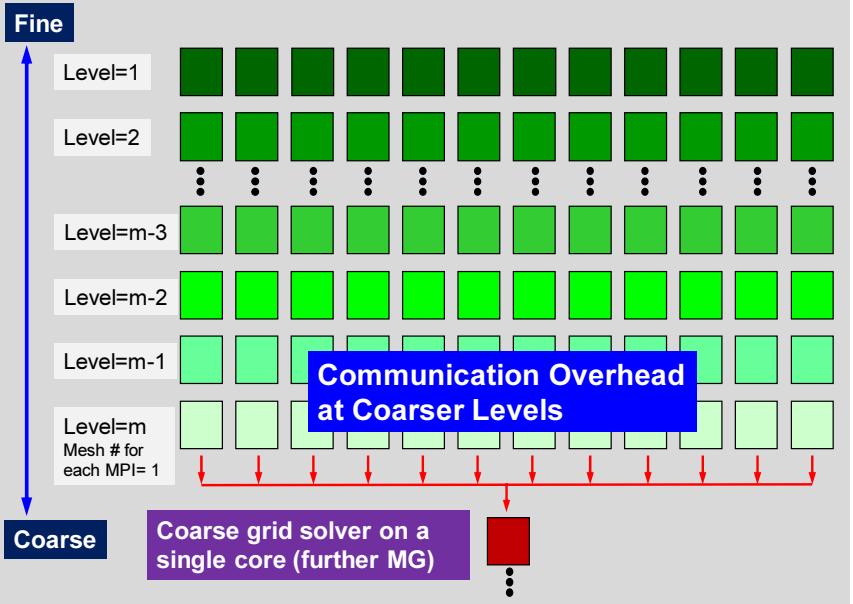
SCS-b (SELL-C-σ) Column-wise

(c) SELL-8-8/Column-Wise (SCS-b)

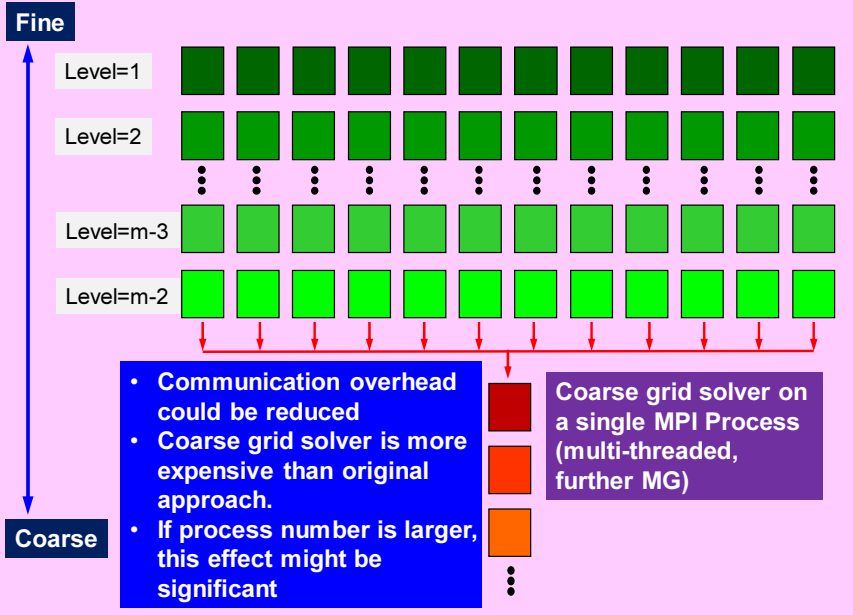
```
!$omp parallel private (icol,…)
      icol= NCOLORtot
      (Operations)
!$omp do
      do (Loops:Threads: ip= 1, PEsmpTOT)
      do (Loops:Blocks)
!$omp simd
        do (Loops SIMD: k= 1, 8)
          (Operations:Substitutions-1)
        enddo
      do (Loops:Non-Zero Off-Diag' s: j= 1, 6)
!$omp simd
        do (Loops SIMD: k= 1, 8)
          (Operations)
        enddo
      enddo
!$omp simd
        do (Loops SIMD: k= 1, 8)
          (Operations:Substitutions-2)
        enddo
      enddo
      enddo
!$omp end parallel
```

More Expensive !!

Parallel Multigrid Methods

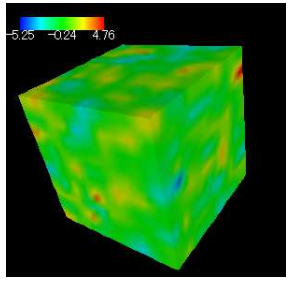


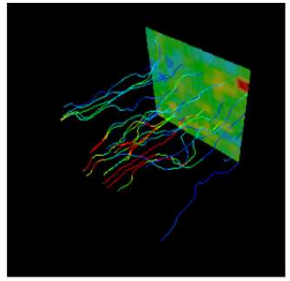
Original Approach [KN 2010]



Coarse Grid Aggregation (CGA) [KN 2012]

Summary of Configurations

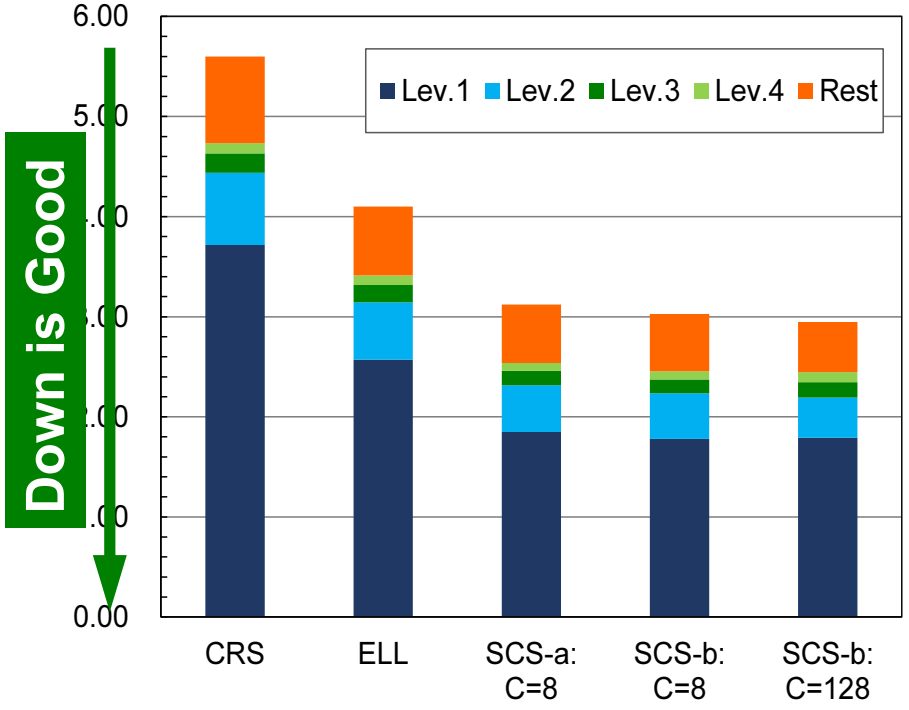




- pGW3D-FVM, Weak Scaling
  - CGA（Coarse Grid Aggregation）: Single Level
- Oakforest-PACS (OFP), ~2,048 nodes
  - Flat, MC-DRAM only
  - 64-cores on each node
    - HB 4x16 (4-threads x 16-proc's), HB 8x8
  - Problem Size: 64x32x32 on each core (max: 8,589,934,592 DOF), Best for 5-Runs, $\varepsilon=10^{-12}$
- Comparison between (CRS, Sliced ELL) and SCS (SELL-C-σ, C=σ=8)
  - 64-bit (Double Precision) x 8 = 512 bit
  - SCS: Switching to Sliced-ELL if the problem size is smaller than (C (=8)) for each color/thread
  - **Sliced ELL for Coarse Grid Solver of SCS**
  - 20% improvement is expected by SCS over Sliced ELL, based on preliminary results

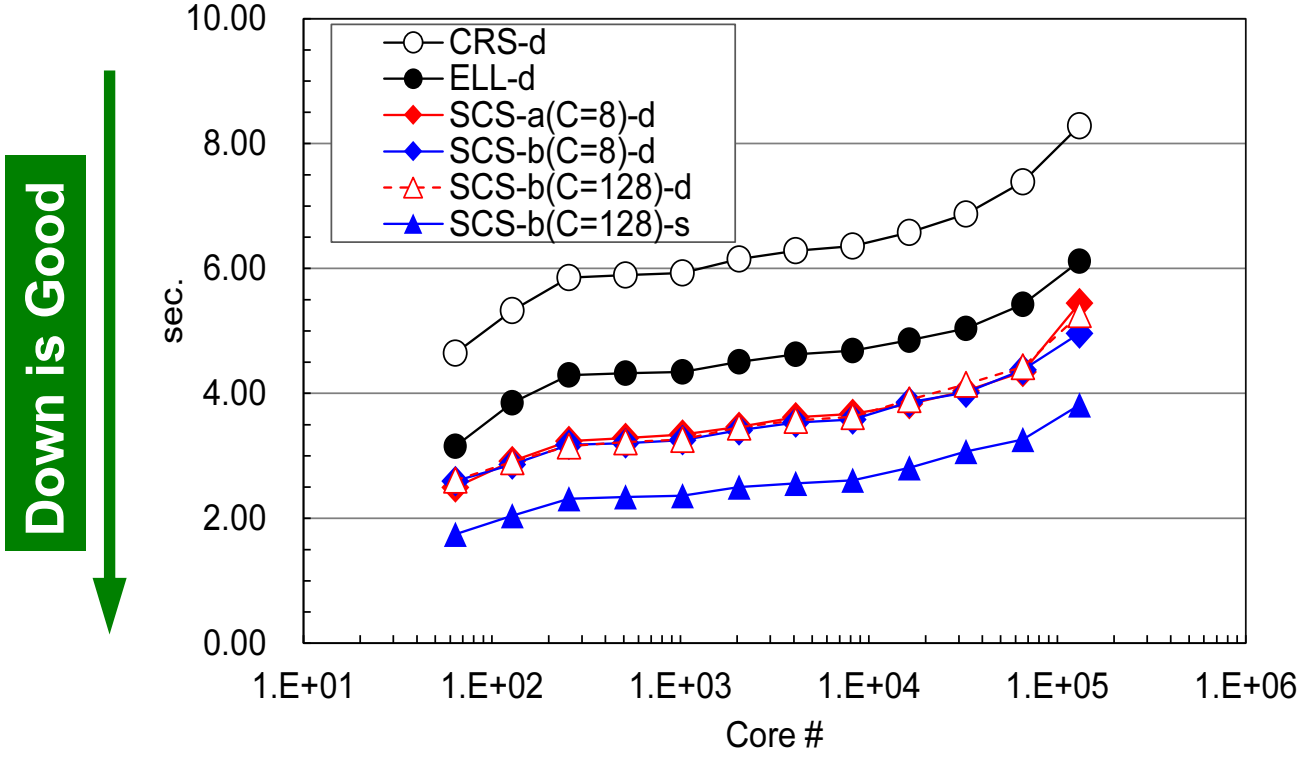
Results: Time for MGCG

8-nodes, HB 4×16, 33,554,432 DOF

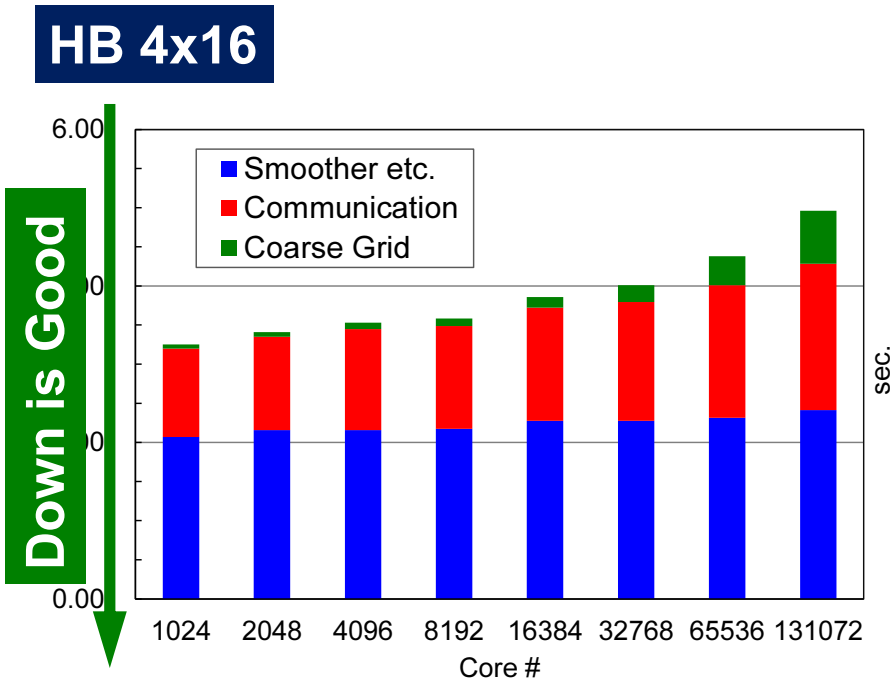


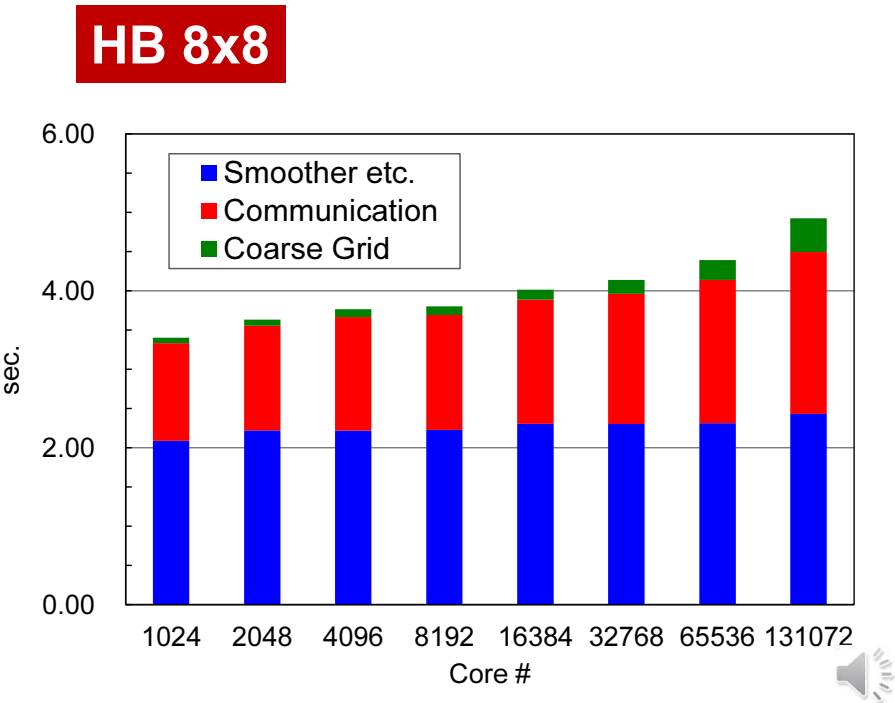
- Each Level of Smoothing
- Rest
  - Coarser Level Smoothers
  - CG except MG (SpMV etc.)
  - Communication
  - Coarse Grid Solver
- Improvement over CRS
  - Sliced ELL：36.6%
  - SCS-a：79.4%
  - SCS-b：84.9% (C=8)
  - SCS-b: 90.1% (C=128)
- Level-1 (Finest)
  - 46.6%, 101.3%, 108.7%, 107.8%
- Sliced ELL⇒SCS
  - SCS-a：31.4%
  - SCS-b：35.4% (C=8)
  - SCS-b: 39.2% (C=128)

Weak Scaling: up to 2,048-nodes
HB 4x16, Time for MGCG, Down is Good



Weak Scaling: up to 2,048-nodes
SCS-b (C=σ=8), Time for MGCG, Down is Good





- Background & Overview
  - Multigtrid Methods
  - Overview of Previous/Present Works
- SELL-C-σ with Double Precision Computing
- **Computing in Double/Single Precision**
- Accuracy Verification
- Summary

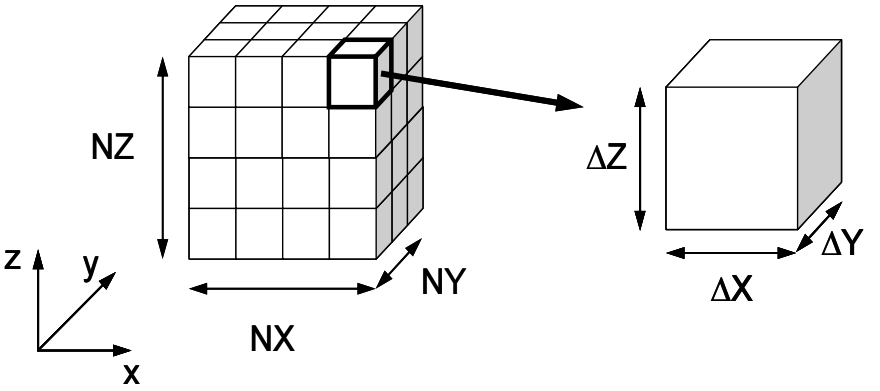
Approximate Computing with Low/Adaptive/Trans Precision

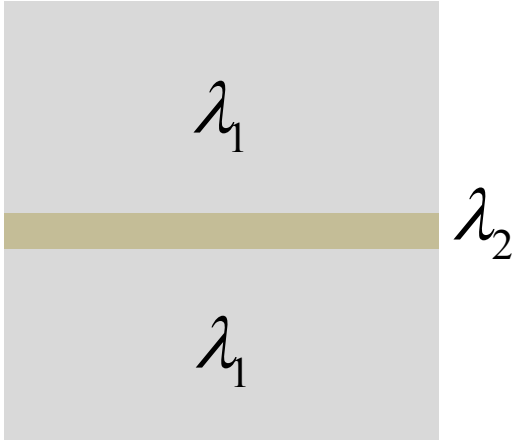
- Mostly, scientific computing has been conducted using FP64 (double precision, DP)
  - Sometimes, problems can be solved by FP32 (single precision, SP) or lower precision
- **Lower precision may save time, energy and memory**
- Approximate Computing
  - Originally for image recognition etc. where accuracy is not necessarily required
  - Also applied to numerical computations
- Computations by lower precision and by mixed precision may provide results with less accuracy

P3D: Steady State 3D Heat Conduction by FVM

$$\nabla \cdot \left(\lambda \nabla \phi \right) + f = 0$$

- 7-point Stencil
- Heterogenous Material Property
  - λ_1/λ_2 is proportional to the condition number of coefficient matrices
- Coefficient Matrix
  - Sparse, SPD
- ICCG Solver
- Fortran 90 + OpenMP
- CM-RCM Reordering
- FP64 (Double), FP32 (Single), FP16 (Half) (just for preconditioning)



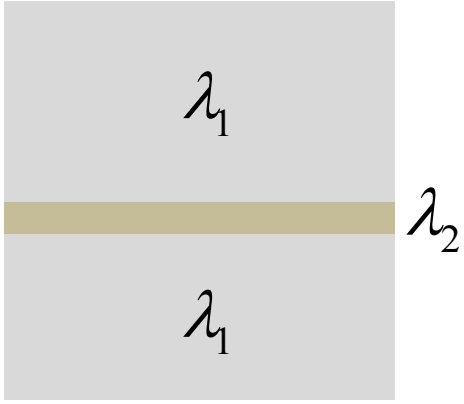


Ratio of FP32(SP)/FP64(DP)

Iterations● & Time△ for ICCG

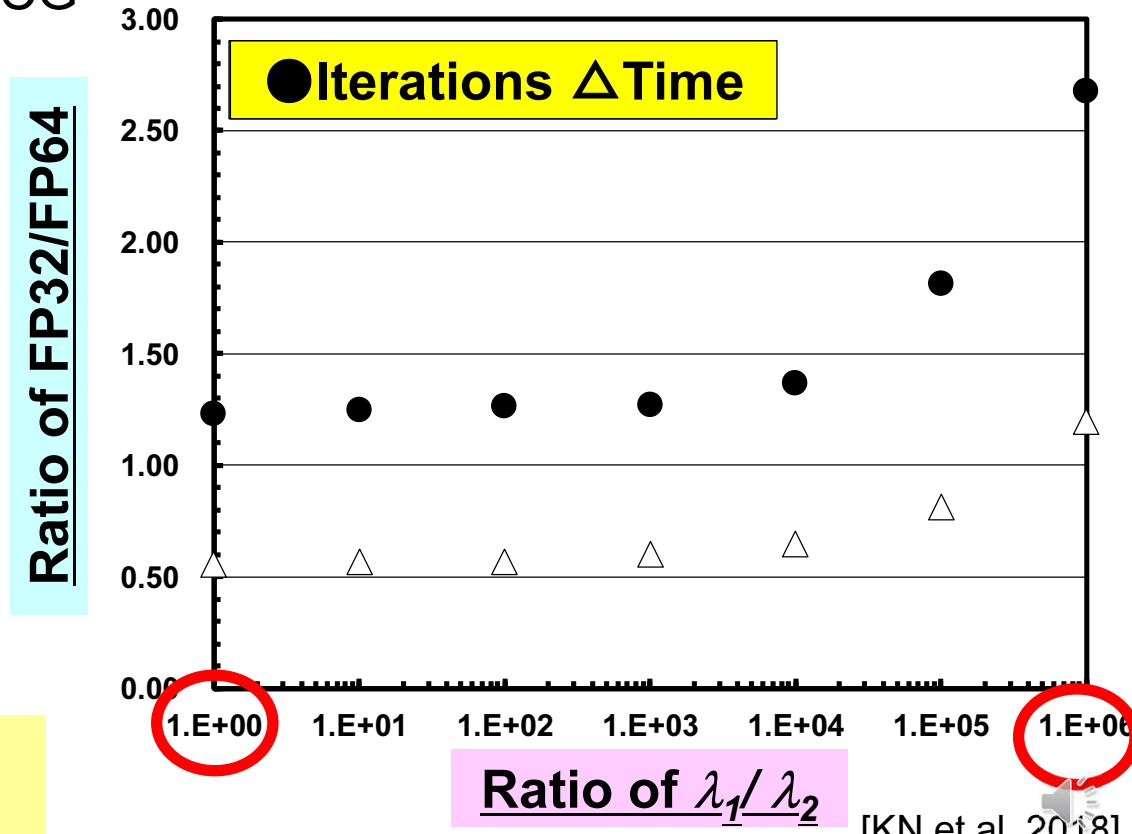
λ_1/λ_2, 128^3 DOF, CRS

Ratio<1 ⇒ FP32 is faster



$$\nabla \cdot (\lambda \nabla \phi) + f = 0$$

Intel Xeon BDW
1 Node: 18 cores x 2 soc's



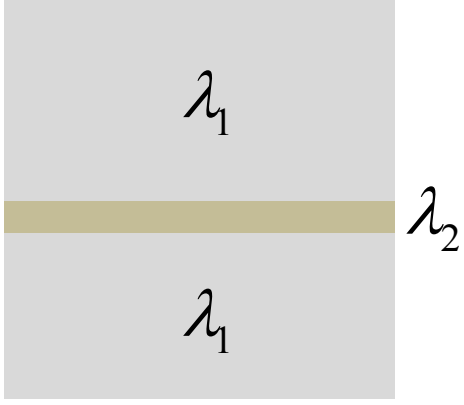
[KN et al. 2018]

Ratio of FP32(SP)/FP64(DP)

Iterations● & Time△ for ICCG

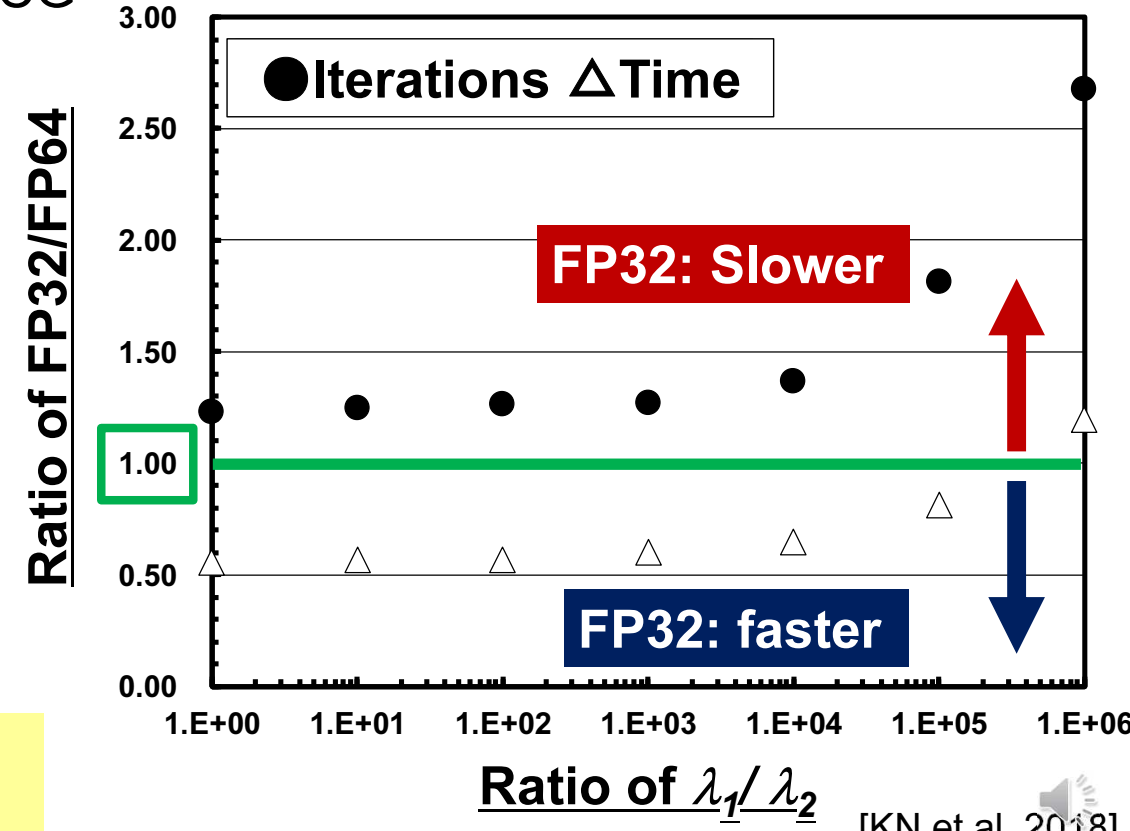
λ_1/λ_2, 128^3 DOF, CRS

Ratio<1 ⇒ FP32 is faster



$$\nabla \cdot (\lambda \nabla \phi) + f = 0$$

Intel Xeon BDW
1 Node: 18 cores x 2 soc's



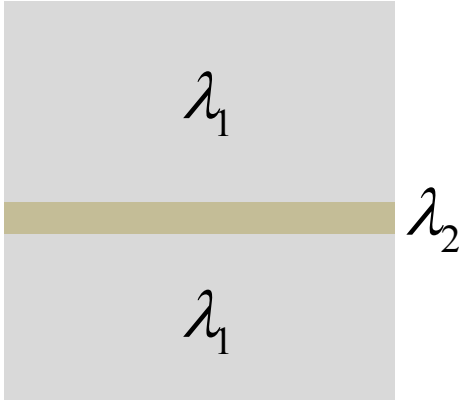
[KN et al. 2018]

Ratio of FP32(SP)/FP64(DP)

Iterations● & Time△ for ICCG

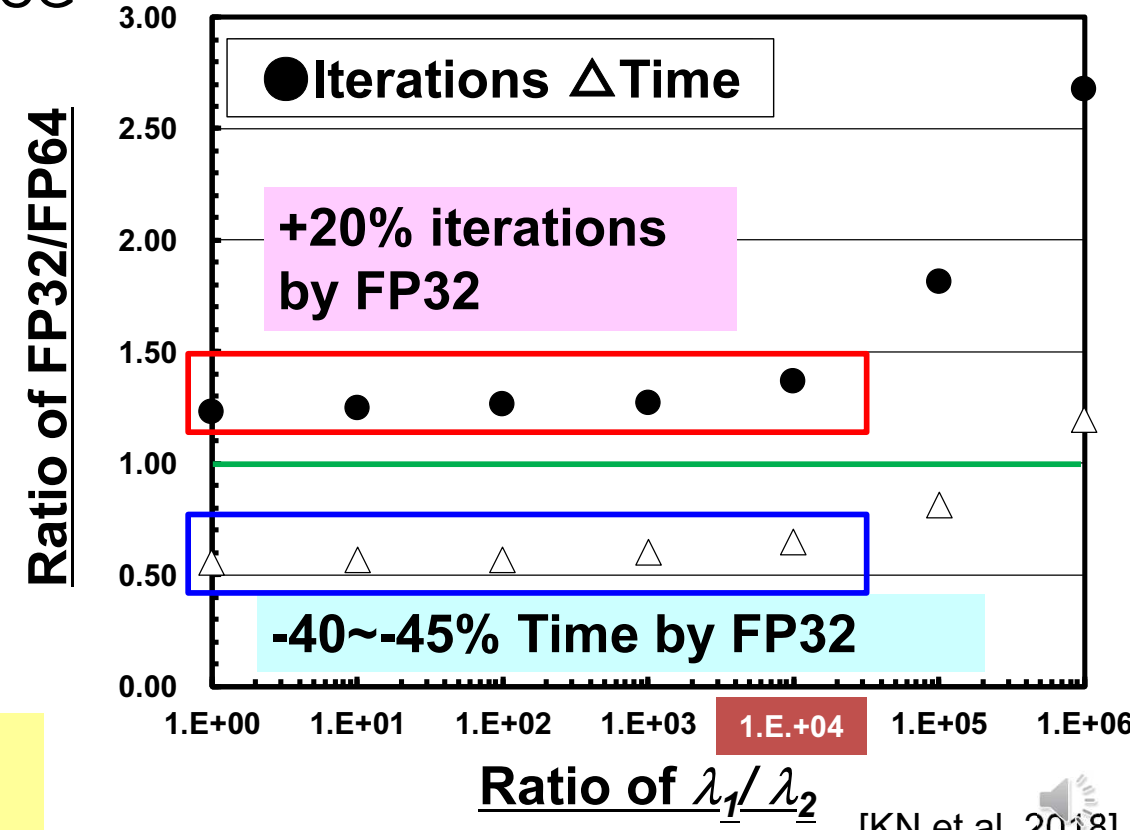
λ_1/λ_2, 128^3 DOF, CRS

Ratio<1 ⇒ FP32 is faster



$$\nabla \cdot (\lambda \nabla \phi) + f = 0$$

Intel Xeon BDW
1 Node: 18 cores x 2 soc's



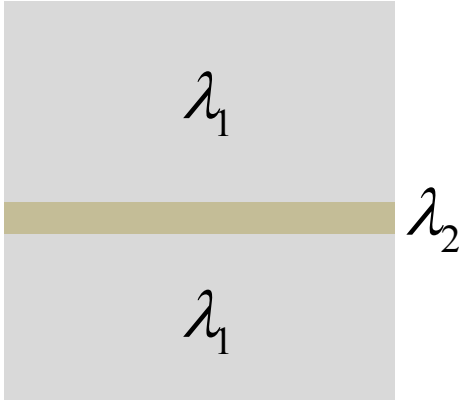
[KN et al. 2018]

Ratio of FP32(SP)/FP64(DP)

Iterations● & Time△ for ICCG

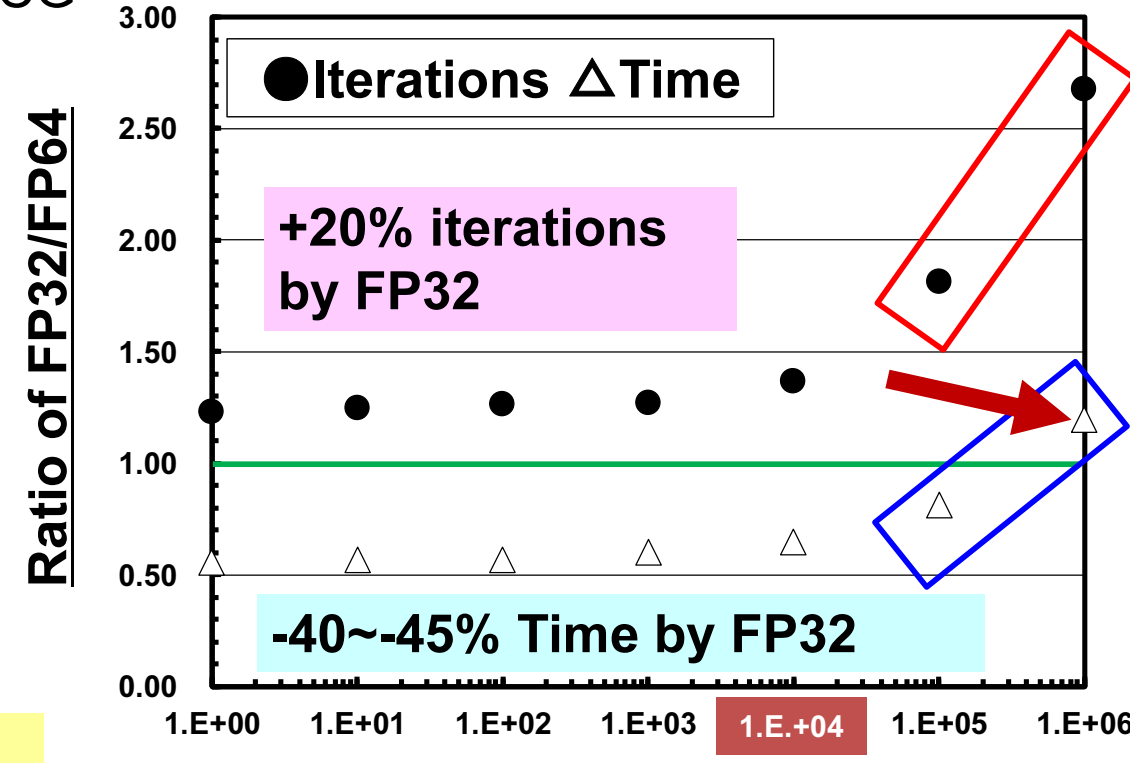
λ_1/λ_2, 128^3 DOF, CRS

Ratio<1 ⇒ FP32 is faster



$$\nabla \cdot (\lambda \nabla \phi) + f = 0$$

Intel Xeon BDW
1 Node: 18 cores x 2 soc's

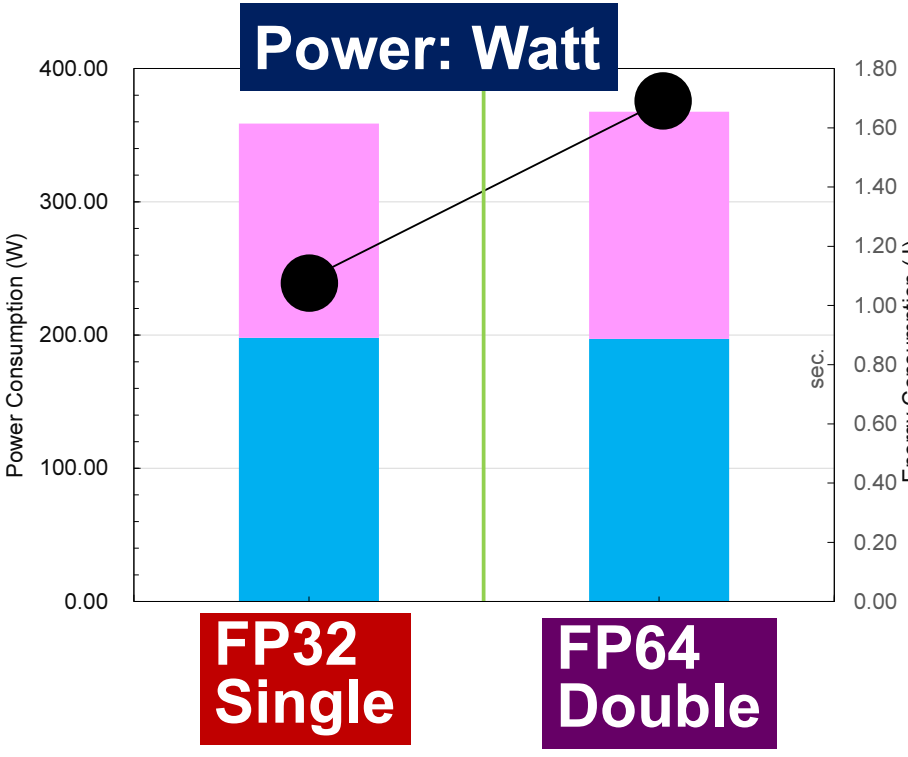


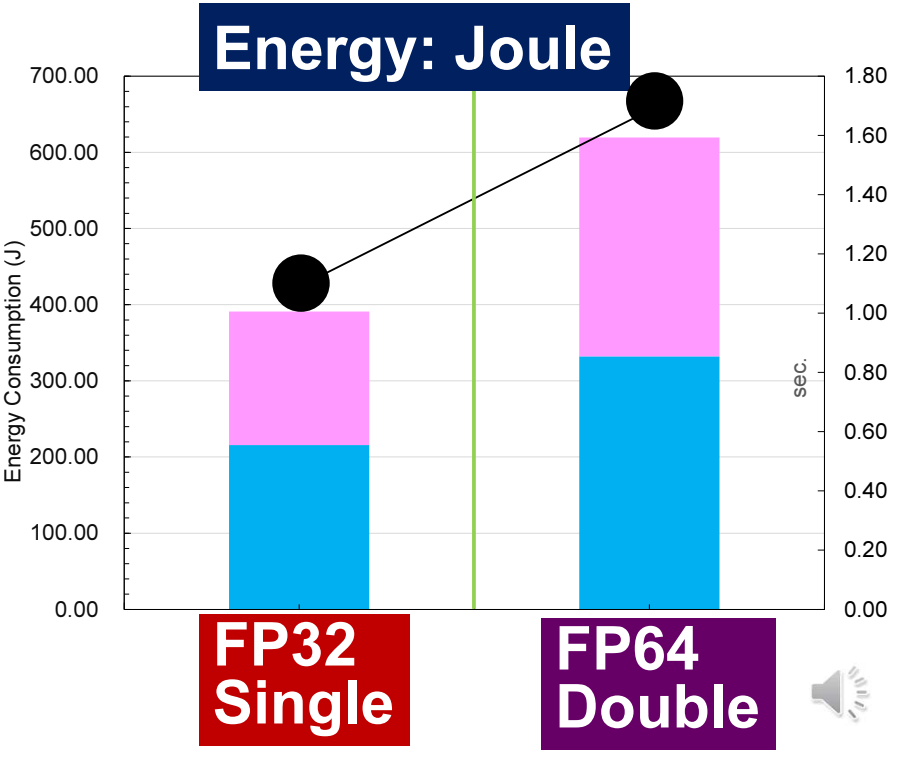
[KN et al. 2018]

Results on Intel Xeon BDW $\lambda_1 = \lambda_2$

[Sakamoto et al. 2020]

N=128³, : CPU, : Memory, : Time





Target Platforms

- Oakforest-PACS (OFP)
  - Intel Xeon Phi (Knights Landing, KNL), Fujitsu
  - IHK/McKernel
  - 8,208 nodes, 25+PF, 22th in TOP 500 (Nov.2020)
  - Operated by JCAHPC (U.Tsukuba & U.Tokyo)
- Oakbridge-CX (OBCX)
  - Intel Platinum 8280 (Cascade Lake, CLX), Fujitsu
  - 1,368 nodes (2,736 sockets), 6.61 PF, 69th in TOP 500 (Nov.2020)

JCAHPC

東京大学 THE UNIVERSITY OF TOKYO

筑波大学 University of Tsukuba

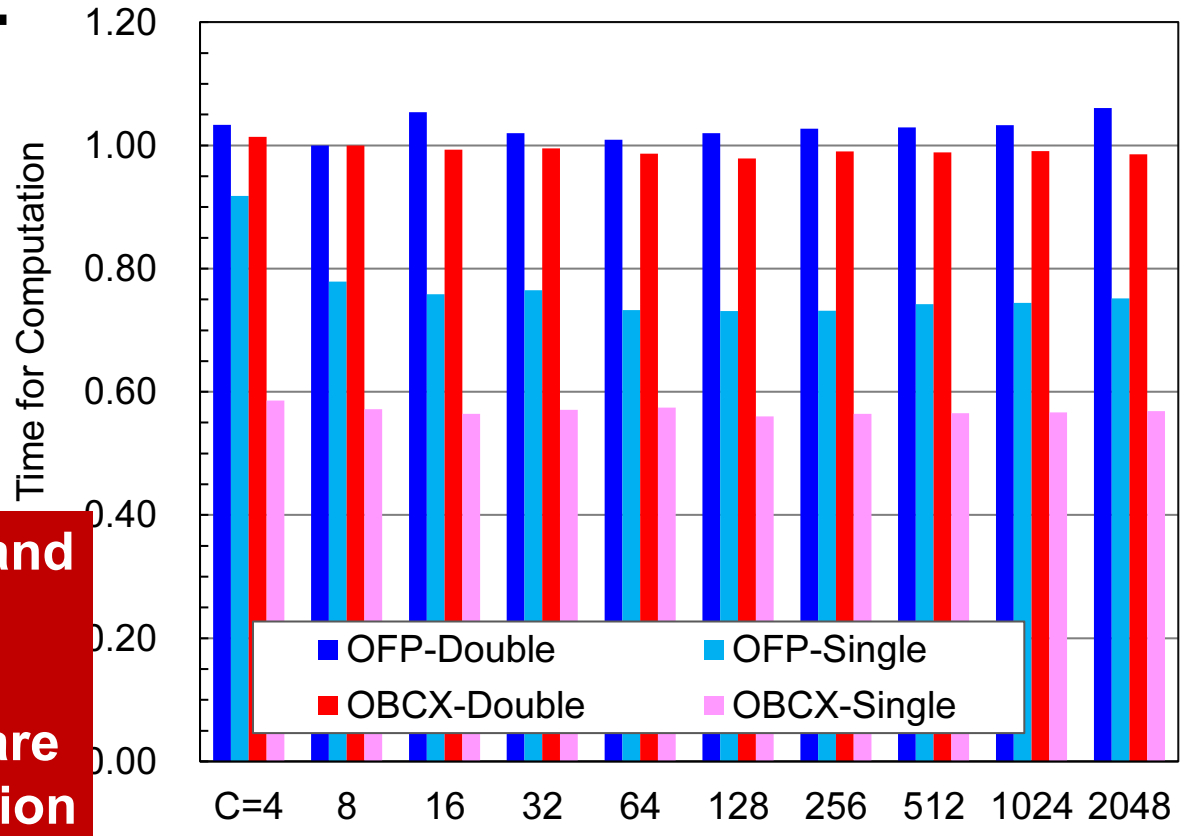
Overview of Each Node (OFP & OBCX)

System	Oakforest-PACS (OFP)	Oakbridge-CX (OBCX)
Name in this Paper	OFP	OBCX
Architecture	Intel Xeon Phi 7250 (Knights Landing, KNL)	Intel Xeon Platinum 8280 (Cascade Lake, CLX)
Frequency (GHz)	1.40	2.70
Core #/CPU (socket)	68	28
CPU (socket) # per node	1	2
Peak Performance (GFLOPS) per node	**3,046.4**	**4,838.4**
Memory Size (GB) per node	MCDRAM: 16 DDR4: 96	192
Memory Bandwidth/Socket (GB/sec, STREAM Triad)	**MCDRAM: 490** **DDR4: 84.5**	**202.0**
Peak Performance per Core (GFLOPS)	44.8	86.4
Memory Bandwidth per Core (GB/sec., STREAM Triad)	MCDRAM: 7.21 DDR4: 1.24	3.61

Time for MGCG: SCS-b (C=σ) Normalized by SELL-8-8 (DP)

8-nodes
OFP(HB 4x16),
OBCX (6x8)

All variables, vectors and matrices are stored in double precision for FP64, and all of them are stored in single precision for FP32

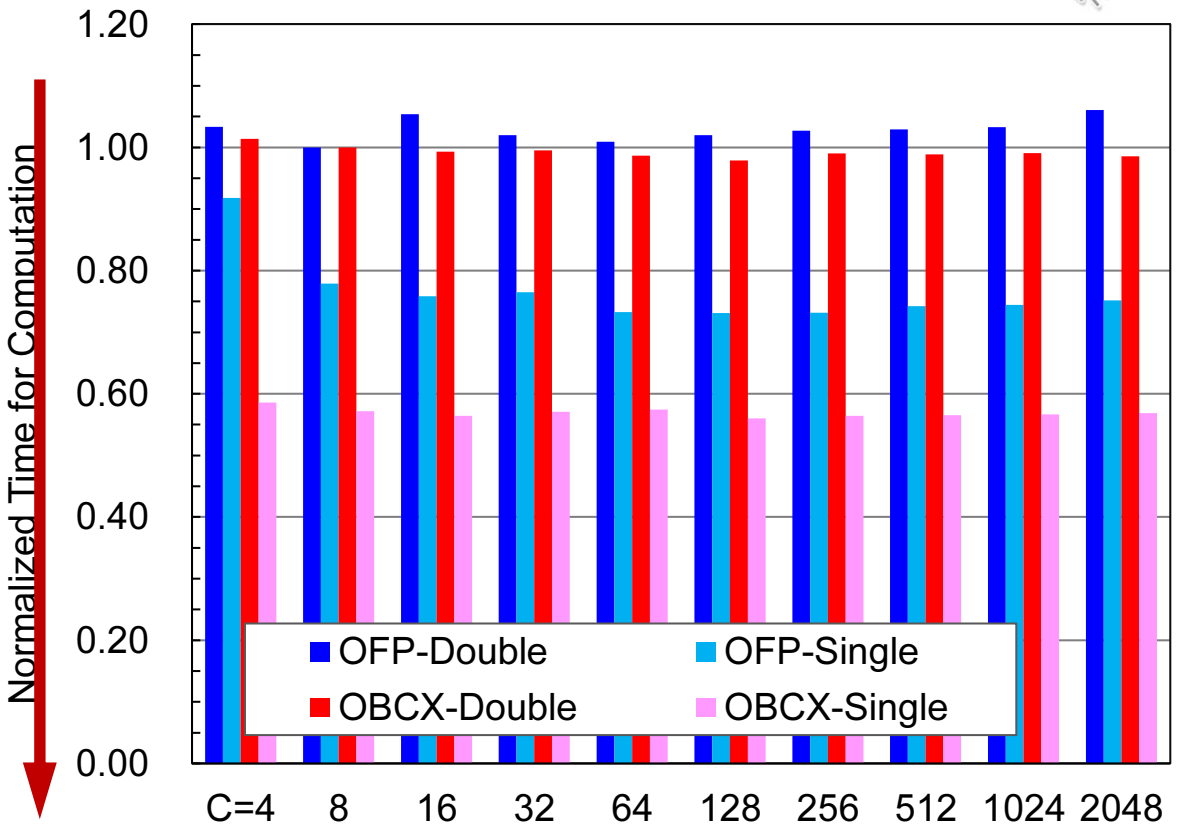


Time for MGCG: SCS-b (C=σ) Normalized by SELL-8-8 (DP)

8-nodes
OFP(HB 4x16),
OBCX (8x8)

Optimum Parameters

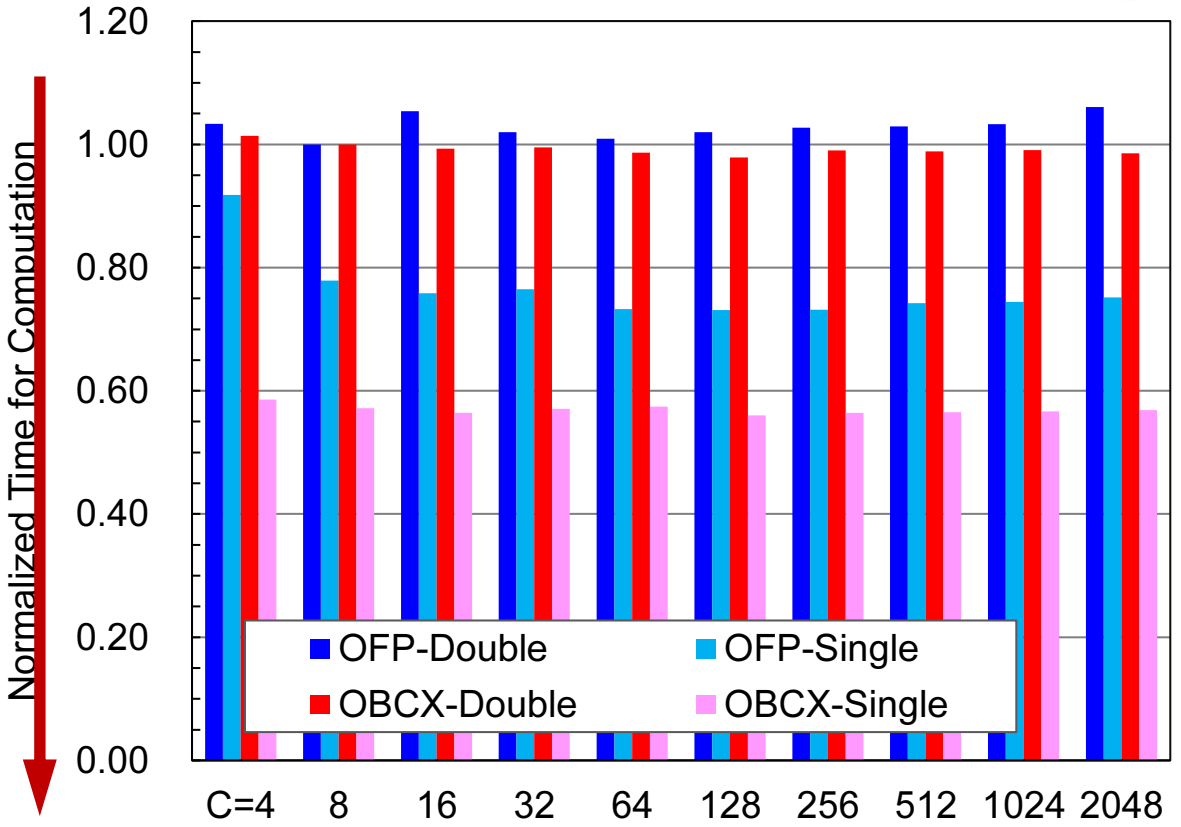
- DP: C=σ= 8
- SP: C=σ=128



Down is Good

DP⇒SP

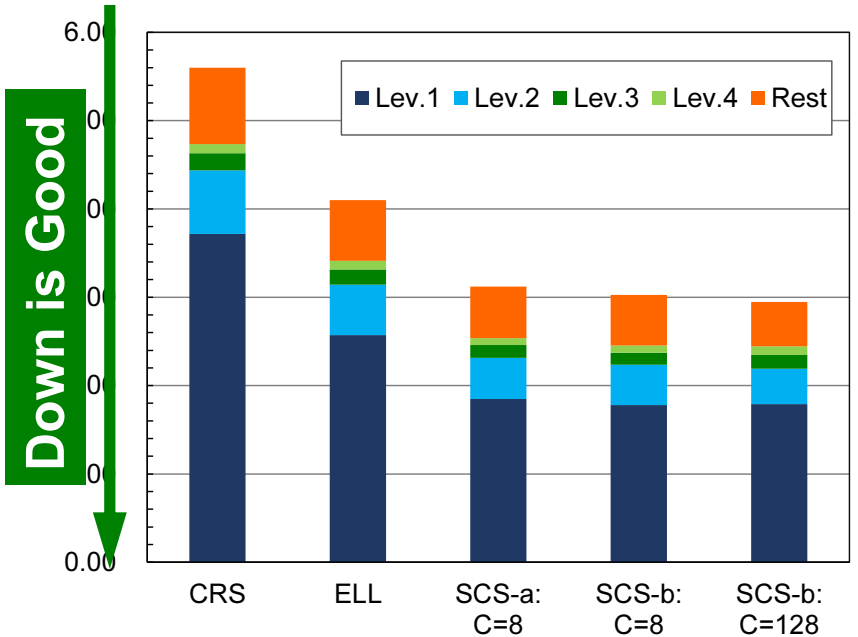
- Iteration number does not change
- <0.1% relative error
- (SP/DP) time ratio
  - 0.70 for OFP
  - 0.55-0.60 for OBCX



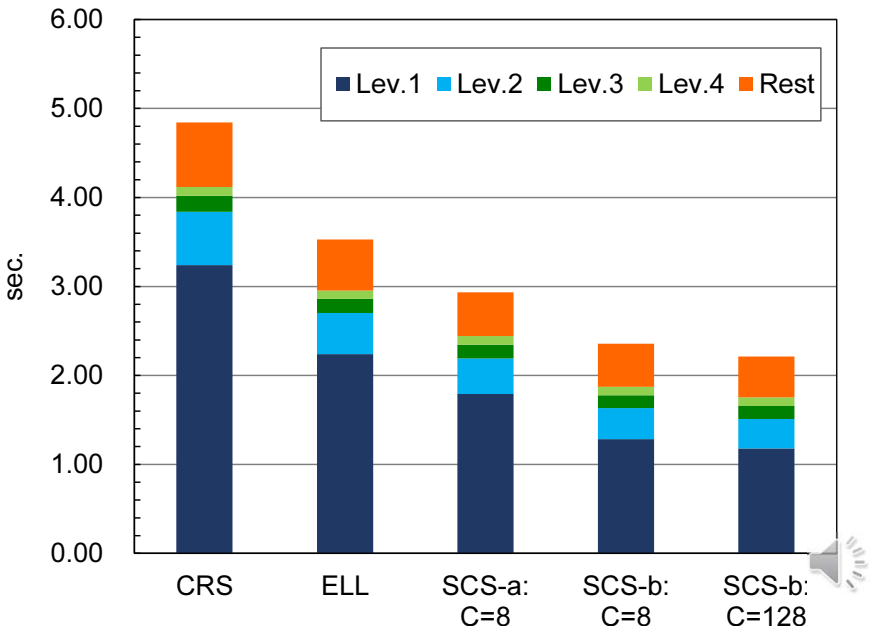
Down is Good

Results: Time for MGCG 8-nodes, OFP (HB 4×16)

FP64, Double



FP32, Single



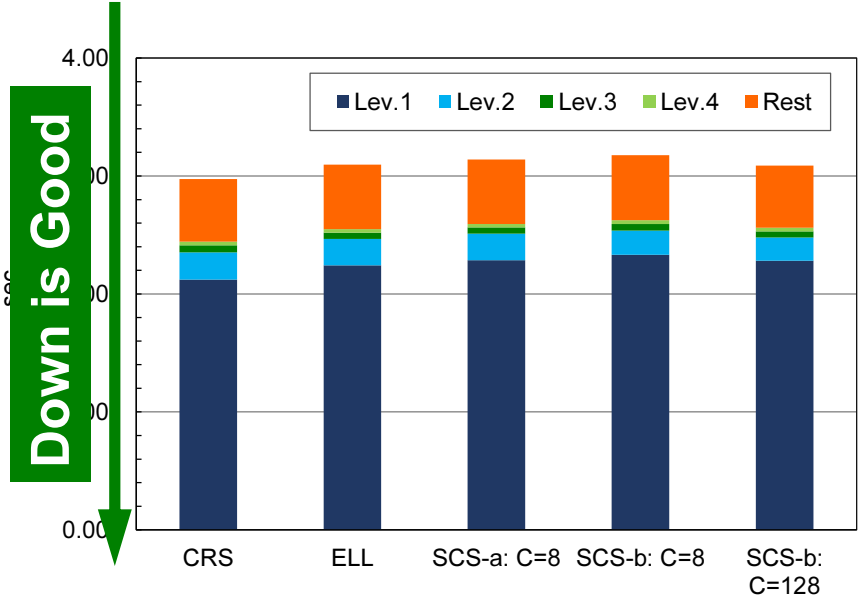
Time for MGCG, 8-nodes, OFP (HB 4×16)

Improvement over CRS (MGCG solver, Level-1 of Smoother)

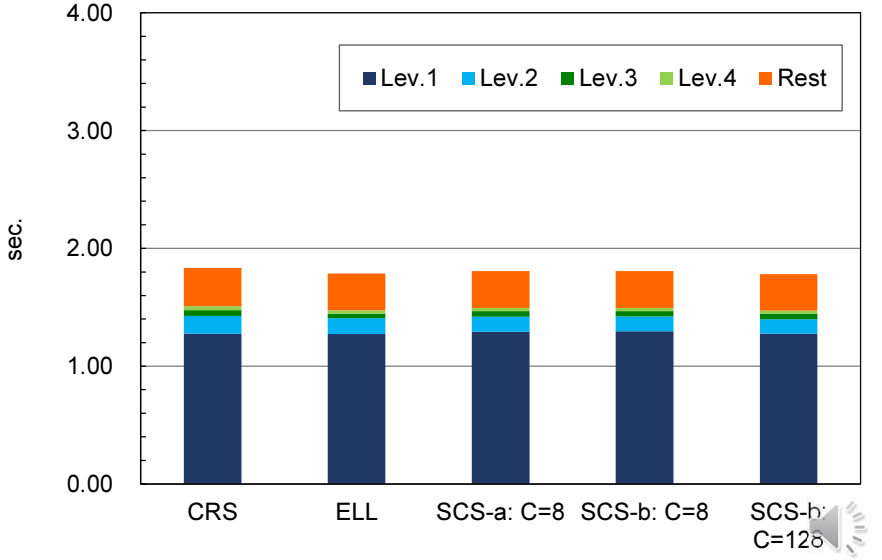
	Double Precision (FP64)	Single Precision (FP32)
Sliced ELL	36.6%, 46.6%	15.6%, 44.7%
SCS-a (C=σ=8)	79.4%, 101.3%	58.7%, 81.0%
SCS-b (C=σ=8)	84.9%, 108.7%	90.9 %, 152.2%
SCS-b (C=σ=128)	90.1%, 107.8%	**137.5%, 174.9%**

Results: Time for MGCG
8-nodes, OBCX (HB 6×8), Down is Good

FP64, Double

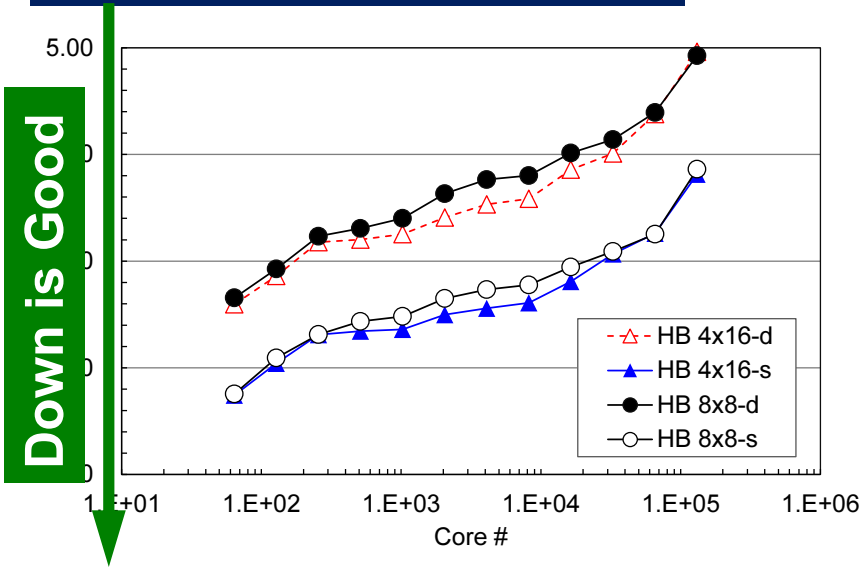


FP32, Single



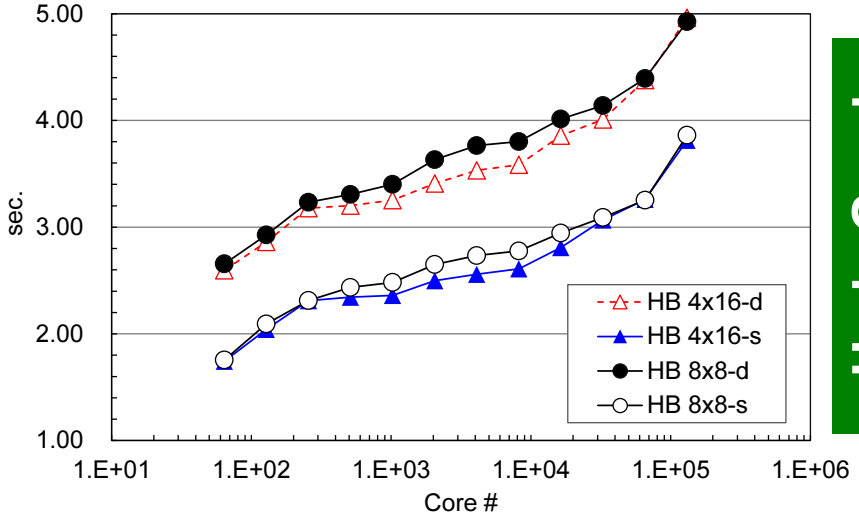
Weak Scaling: up to 2,048-nodes of OFP (DP: C=σ=8, SP: C=σ=128)

Time for MGCG: SCS-b
DP: C=σ=8, SP: C=σ=128



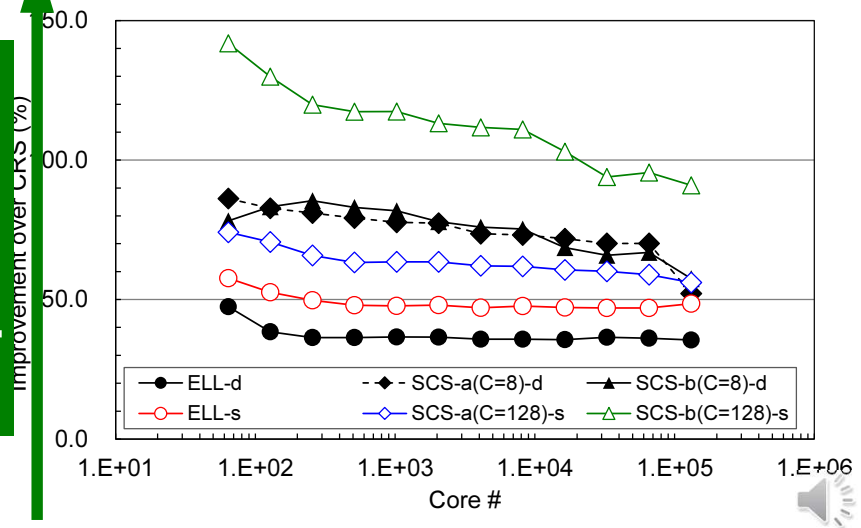
Weak Scaling: up to 2,048-nodes of OFP (DP: C=σ=8, SP: C=σ=128)

Time for MGCG: SCS-b
DP: C=σ=8, SP: C=σ=128



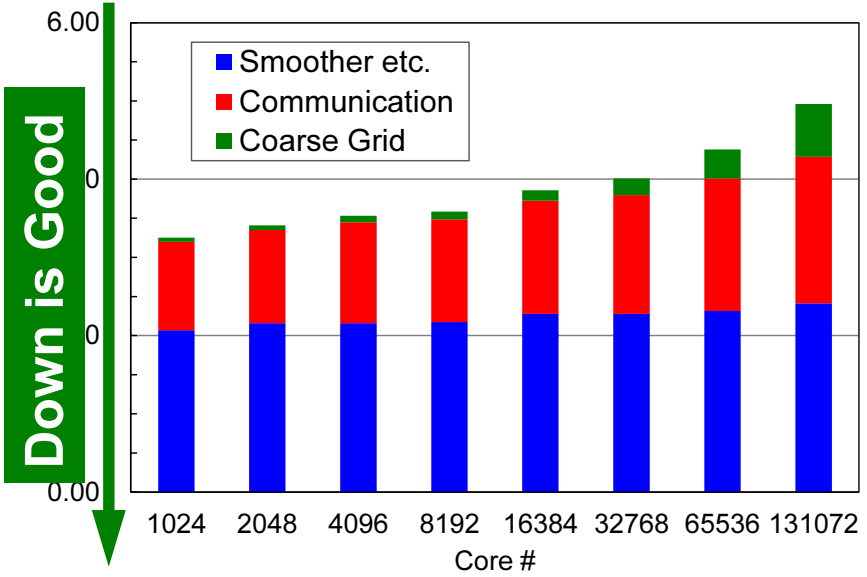
Improvement over CRS

Up is Good

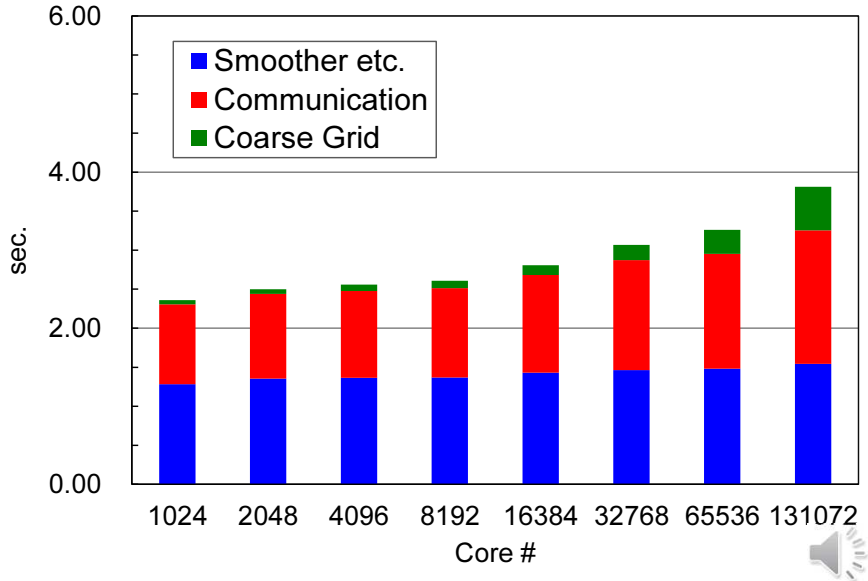


Weak Scaling: up to 2,048-nodes of OFP, HB 4x16, Time for MGCG, Down is Good

FP64: C=σ=8



FP32: C=σ=128



- Background & Overview
  - Multigtrid Methods
  - Overview of Previous/Present Works
- SELL-C-σ with Double Precision Computing
- Computing in Double/Single Precision
- **Accuracy Verification**
- Summary

Approximate Computing with Low/Adaptive/Trans Precision

- Accuracy verification is important, especially for computation with lower/mixed precision.
- A lot of methods for accuracy verification have been developed for problems with dense matrices
  - But very few examples for sparse matrices & H-matrices
- Generally speaking, processes for accuracy verification is very expensive
  - Sophisticated Method needed
  - Automatic Selection of Optimum Precision by Technology of AT (Auto Tuning)
- Accuracy Verification of Sparse Linear Solvers [Ogita, Nakajima 2019]

New Algorithm for Verification
[Ogita, Rump, Oishi 2005] [Ogita, Nakajima 2019]

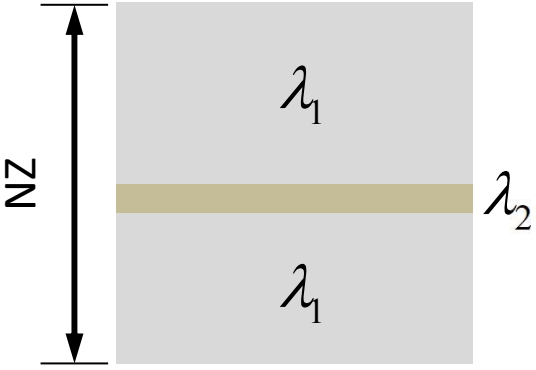
1. Solve a discretized linear system $Ax = b$.
2. Solve a linear system $Ay = e$.
3. Verify M-property of A using $\hat{y}$. $(\hat{y} > 0 \;\Rightarrow\; A\hat{y} > 0)$
4. Compute $r = b - A\hat{x}$ with an error bound.
   - $\hat{r}$: a computed residual, e_r: an error bound of $\hat{r}$
5. Solve a linear system $Az = \hat{r}$.
6. Compute an error bound using

$$\|x - \hat{x}\|_\infty \le \|\hat{z}\|_\infty + \frac{\|\hat{y}\|_\infty (\|\hat{r} - A\hat{z}\|_\infty + \|e_r\|_\infty)}{1 - \|e - A\hat{y}\|_\infty}.$$

Results: New Alg. for Verification (128^3)

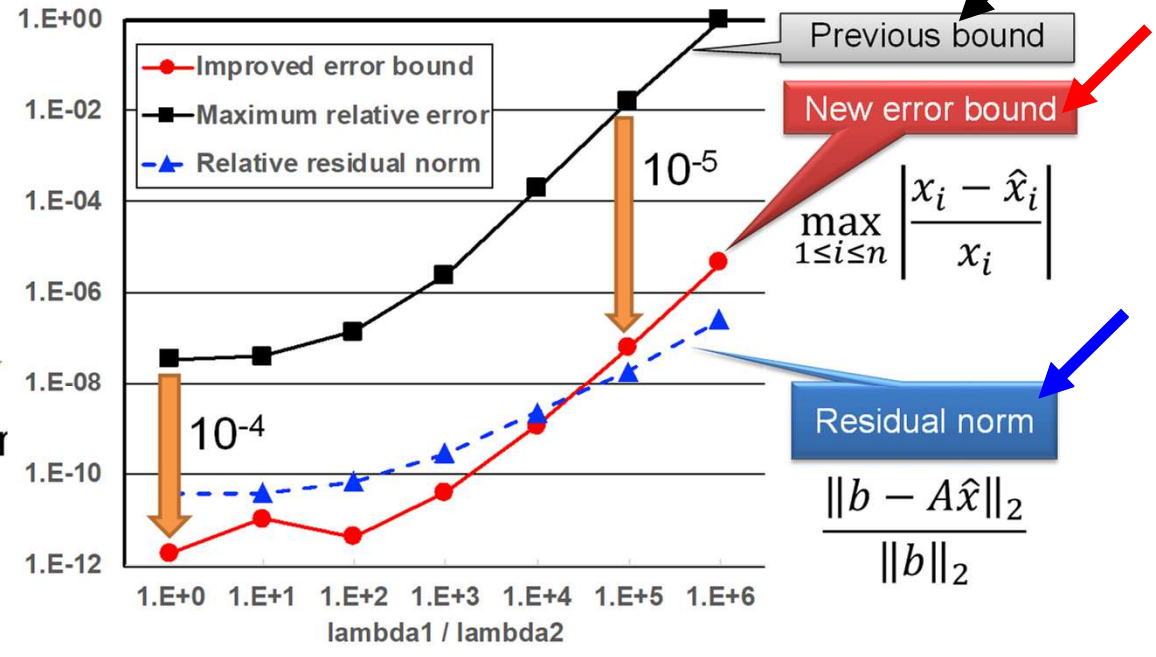
[Ogita, Rump, Oishi 2005] [Ogita, Nakajima 2019]

$$\phi = 0 \,@\, z = z_{\max}$$



$$\nabla \cdot (\lambda \nabla \phi) + f = 0$$

better



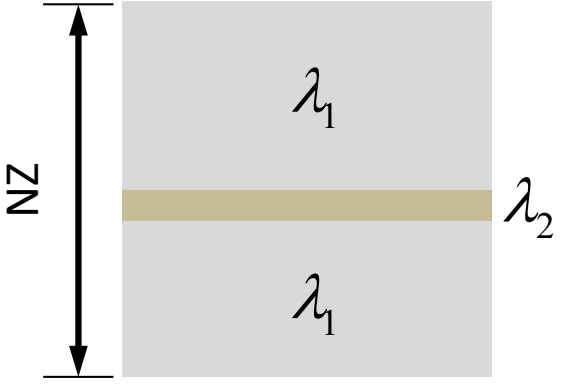
Computed error bounds are significantly improved!

Results on OBCX (Intel Xeon CXL) (1/2)

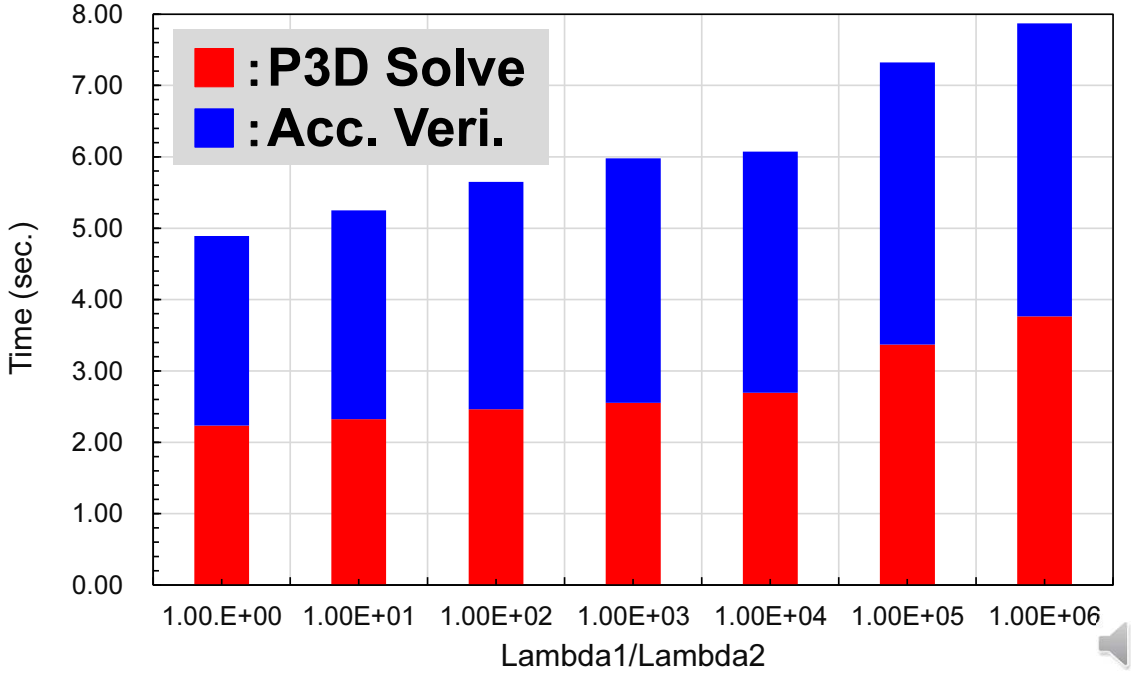
FP64, Accuracy Verification ■ takes 10% longer

λ_1/λ_2 =10^0~10^6, CRS, N=128^3

$$\phi = 0 @ z = z_{\max}$$



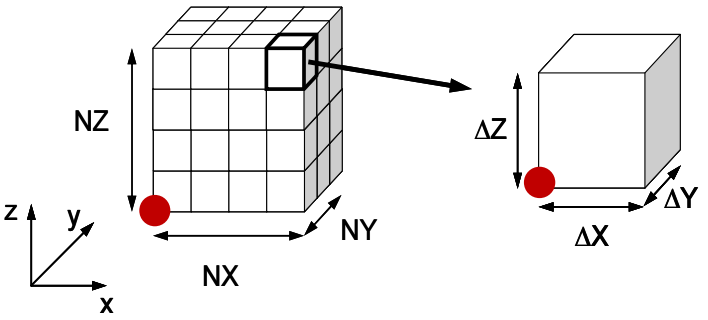
$$\nabla \cdot (\lambda \nabla \phi) + f = 0$$



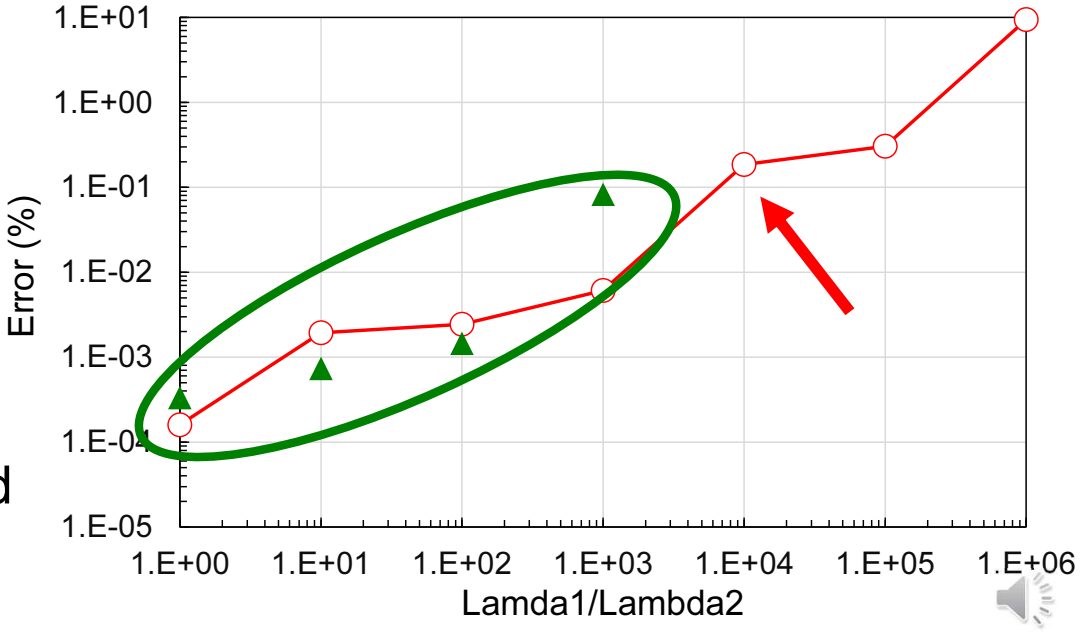
Results on OBCX (2/2)

Accuracy verification for FP32 has failed, if $\lambda_1/\lambda_2 \geqq 10^4$

$$\left\|x-\hat{x}\right\|_\infty \leq \left\|\hat{z}\right\|_\infty + \frac{\left\|\hat{y}\right\|_\infty \left\|b-A\left(\hat{x}+\hat{z}\right)\right\|_\infty}{1-\left\|e-A\hat{y}\right\|_\infty}$$

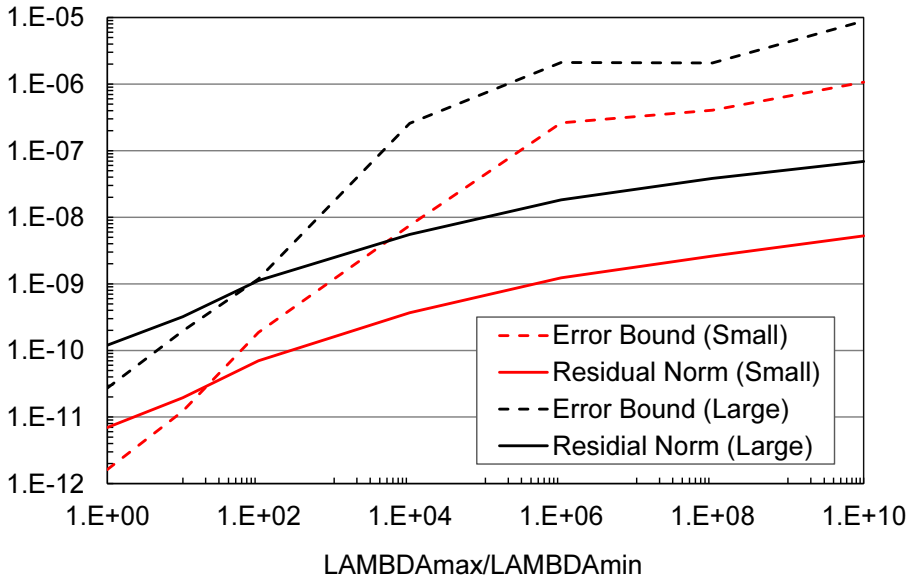


- ○ Relative Error between FP32 & FP64 at ●
- ▲ Max. Relative Error Bound for FP32 obtained from Accuracy Verification



Accuracy Verification in pGW3D-FVM

- 1st Application of the New Verification Method [Ogita, Nakajima 2019] to Distributed Parallel Computing
- Two Cases on OFP (HB 8x8)
  - Small: 128 meshes, 1-node
  - Large: 1,024x1,024x512 meshes, 128 nodes
- DP only (SP failed in Accuracy Verification)



- Background & Overview
  - Multigtrid Methods
  - Overview of Previous/Present Works
- SELL-C-σ with Double Precision Computing
- Computing in Double/Single Precision
- Accuracy Verification
- **Summary**

Summary

- SELL-C-σ with double/single precision computing in the MGCG solver using up to 2,048 nodes of Intel Xeon Phi.
- Improvement of the performance by SELL-C-σ over the sliced ELL
  - 35+% for DP, 45+% for SP on OFP.
  - The effect of SELL-C-σ for computing with SP is very significant
- Accuracy Verification (Preliminary)
- The first example of SELL-C-σ applied to forward/backward substitutions in ILU-type smoother of multigrid solver with double/single precision computing.
- Future Works
  - SELL-C-σ for Coarse Grid Solver
  - Improvement of Accuracy Verification