Communication-Computation Overlapping with Dynamic Loop Scheduling for Preconditioned Parallel Iterative Solvers on Multicore/Manycore Clusters

Kengo Nakajima, Toshihiro Hanawa
Information Technology Center, The University of Tokyo

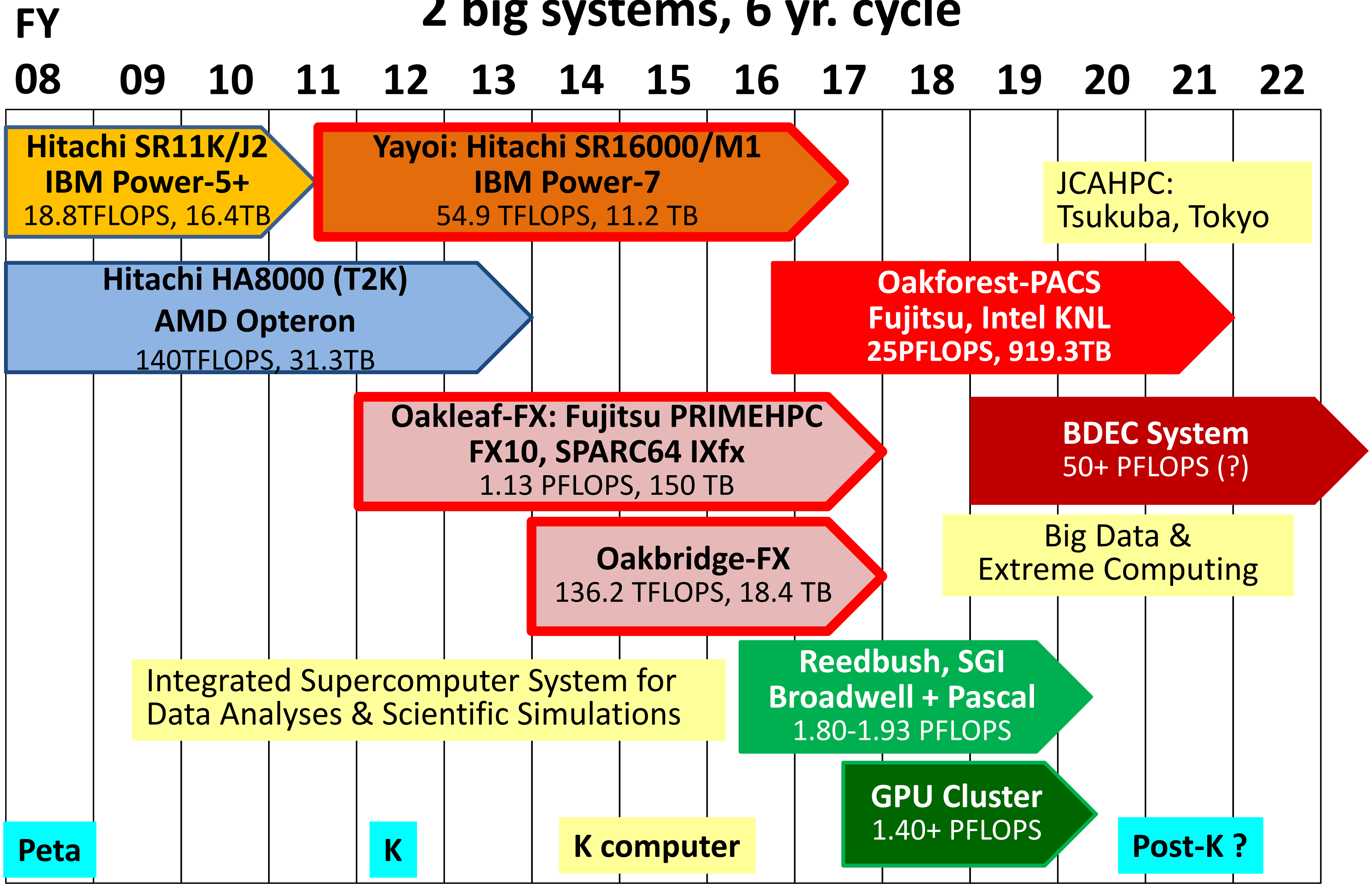
10th International Workshop on Parallel Programming Models & Systems Software for High-End Computing (P2S2 2017)
in conjunction with the 46th International Conference on Parallel Processing (ICPP 2017)
August 14, 2017, Bristol, UK

Acknowledgements

- JST-CREST, Japan
  - ppOpen-HPC Project
- DFG-SPPEXA, Germany
  - ESSEX-II Project
- JCAHPC (Joint Center for Advanced High Performance Computing)
  - CCS, University of Tsukuba
  - ITC, The University of Tokyo
- Prof. Scott Baden (LBNL)
- Dr. Jack Deslippe (LBNL)

Supercomputers in ITC/U.Tokyo

2 big systems, 6 yr. cycle



Oakforest-PACS (OFP)

- Full Operation started on December 1, 2016
- 8,208 Intel Xeon/Phi (KNL), 25 PF Peak Performance
  - Fujitsu
- **TOP 500 #7 (#1 in Japan), HPCG #5 (#2) (June 2017)**
- **JCAHPC: Joint Center for Advanced High Performance Computing)**
  - 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
  - http://jcahpc.jp

Features of Oakforest-PACS

- Computing Node
  - 68 cores/node, 3 TFLOPS x 8,208= 25 PFLOPS
  - 2 Types of Memory
    - MCDRAM: High-Speed, Large-Latency, 16GB
    - DDR4: Medium-Speed, 96GB
  - Variety of Selections for Memory-Mode/Cluster-Mode
- Node-to-Node Communication
  - Fat-Tree Network with Full Bi-Section Bandwidth
  - Intel's Omni-Path Architecture
    - High Efficiency for Applications with Full Nodes of the System
    - Flexible and Efficient Operations for Multiple Jobs

- Introduction
- Dynamic Loop Scheduling
- Hardware Environment

- Preliminary Results by Parallel FEM (GeoFEM/Cube)
- Strong Scaling
- SAI Preconditioning

- Summary

Overview of the Present Work

- Communication-Computation Overlapping (*CC-Overlapping*) in Sparse Matrix-Vector Multiplication (SpMV)
- Dynamic Loop Scheduling of OpenMP
- Performance Evaluation by Parallel FEM Application (GeoFEM/Cube) on multicore/manycore clusters.
- Performance Improvement by 40%-50% for Preconditioned CG Solvers in Strong Scaling up to 16,384 cores of Fujitsu PRIMEHPC FX10 (FX10) and KNL Cluster (Oakforest-PACS, OFP).
- **15%-20% improvement** for GeoFEM/Cube with SAI-BiCGSTAB using 12,288 cores of Fujitsu FX10 and OFP.

Communication/Synchronization Avoiding/Reducing/Hiding

for Parallel Preconditioned Krylov Iterative Methods

- Dot Products
  - Pipelined Methods [Ghysels et al. 2014]
  - Gropp's Algorithm
  - Utilization of asynchronous collective communications (e.g. MPI_Iallreduce) supported in MPI-3 for hiding such overhead.
- SpMV
  - Overlapping of Comp. & Comm. (CC-Overlapping)
  - + Dynamic Loop Scheduling
  - Matrix Powers Kernels [Hoemmen et al. 2010]

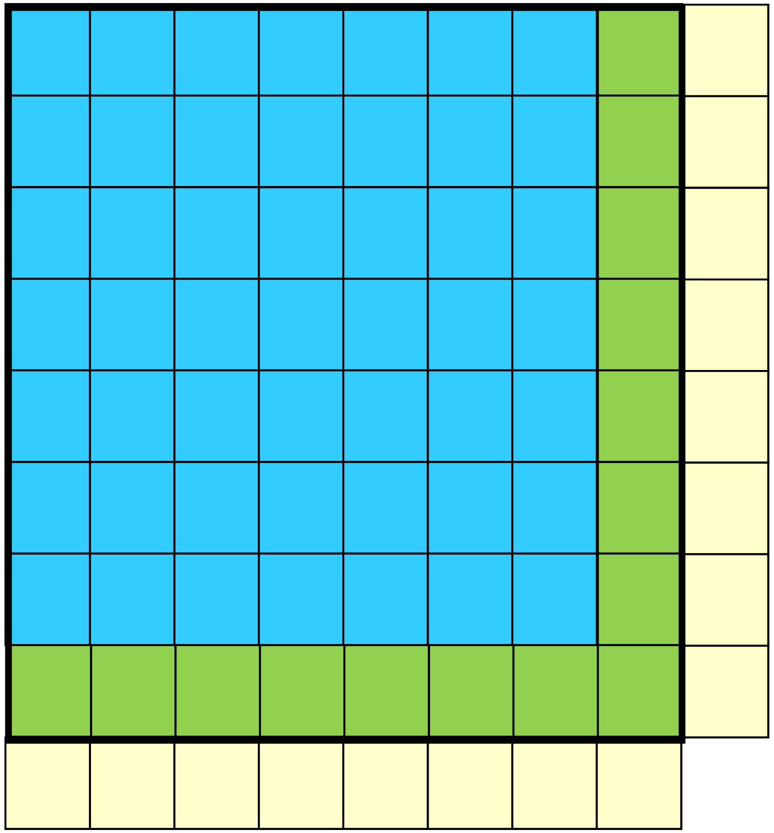
```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    solve [M]z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check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

- Introduction
- **Dynamic Loop Scheduling**
- Hardware Environment

- Preliminary Results by Parallel FEM (GeoFEM/Cube)
- Strong Scaling
- SAI Preconditioning

- Summary

Comm.-Comp. Overlapping (CC-Overlapping): ***Static***



Pure Internal Meshes

External (HALO) Meshes

Internal Meshes on Boundary's (Boundary Meshes)

```
call MPI_Isend
call MPI_Irecv
```

```
do i= 1, Ninn
  (calculations)
enddo
call MPI_Waitall
```

```
do i= Ninn+1, Nall
  (calculationas)
enddo
```

Good for Stencil
Not so Effective SpMV

Dynamic Loop Scheduling (1/2)

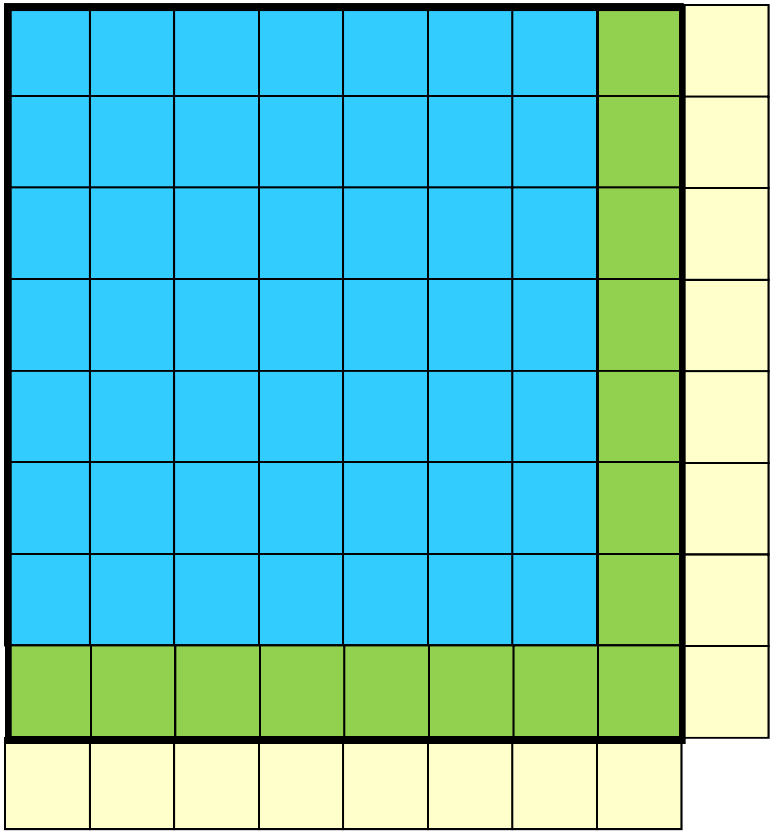
- CC-Overlapping in HALO Exchange
  - HALO exchange including sending buffer copy is done by the *master* thread
  - Dynamic loop scheduling is applied to the computations for *pure* internal nodes/meshes
  - The computations for pure internal nodes/meshes starts without master thread, while the master thread is doing communications.
  - The master thread can join the computations for pure internal nodes/meshes after completion of the communication.
- There are four different loop scheduling types (*kinds*) (*static, dynamic, guided, auto*), and the optional parameter (*chunk*) must be a positive integer:

C: #pragma omp parallel for schedule (kind [, chunk])
Fortran: !$omp parallel do schedule (kind [,chunk])

Dynamic Loop Scheduling (2/2)

- The kind "*static*" is the default, and loops are divided into equal-sized chunks (or as equal as possible)
  - By default, chunk size is loop-count/number-of-threads.
- If the kind "*dynamic*" is applied, the internal work queue is used for giving a chunk-sized block of loop iterations to each thread.
  - When operations of a thread have finished, that retrieves the next block of loop iterations from the top of the work queue.
  - The chunk size is equal to 1 by default.
  - Extra overhead for scheduling is involved for this type of scheduling.
- (Next Page) Pseudo Code with Dynamic Loop Scheduling
  - Global communications are done by the master thread between "!$omp master" and "!$omp end master".
  - The loop for computations of pure inner nodes/meshes with dynamic scheduling starts without the master thread, and that join the loop operations after the completion of communications.
  - Smaller value of *chunk size* may prevent *load imbalance* among threads, but extra operations related to the internal work occur more frequently for smaller *chunk size*, which may lead to very significant overhead.

Comm.-Comp. Overlapping + Dynamic Loop Scheduling: ***Dynamic***



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

Master:
```
call MPI_Isend
call MPI_Irecv
call MPI_Waitall
```

Dynamic:
```
do i= 1, Ninn
   (calculations)
enddo
```

Static:
```
do i= Ninn+1, Nall
   (calculationas)
enddo
```

Dynamic Loop Scheduling

- **"dynamic"**
- **"!$omp master~!$omp end master"**

```
!$omp parallel private (neib,j,k,i,X1,X2,X3,WVAL1,WVAL2,WVAL3)
!$omp&         private (istart,inum,ii,ierr)

!$omp master                Communication is done by the master thread (#0)
!C
!C- Send & Recv.
(…)
      call MPI_WAITALL (2*NEIBPETOT, req1, sta1, ierr)
!$omp end master

!C                            The master thread can join computing of internal
!C-- Pure Internal Nodes      nodes after the completion of communication

!$omp do schedule (dynamic,200)    Chunk Size= 200
      do j= 1, Ninn
        (…)
      enddo
!C
!C-- Boundary Nodes           Computing for boundary nodes are by all threads

!$omp do                      default: !$omp do schedule (static)
      do j= Ninn+1, N
        (…)
      enddo

!$omp end parallel
```

Ina, T., Asahi, Y., Idomura, Y., Development of optimization of stencil calculation on Tera-flops many-core architecture, IPSJ SIG Technical Reports 2015-HPC-152-10, 2015 (in Japanese)

- Introduction
- Dynamic Loop Scheduling
- **Hardware Environment**

- Preliminary Results by Parallel FEM (GeoFEM/Cube)
- Strong Scaling
- SAI Preconditioning

- Summary

3 of 5 used for the present work


- Yayoi (Hitachi SR16000, IBM Power7)
  - 54.9 TF, Nov. 2011 – Oct. 2017
- Oakleaf-FX (Fujitsu PRIMEHPC FX10)
  - 1.135 PF, Commercial Version of K, Apr.2012 – Mar.2018
- Oakbridge-FX (Fujitsu PRIMEHPC FX10)
  - 136.2 TF, for long-time use (up to 168 hr), Apr.2014 – Mar.2018
- Reedbush (SGI, Intel BDW + NVIDIA P100 (Pascal))
  - Integrated Supercomputer System for Data Analyses & Sc Simulations
  - 1.93 PF, Jul.2016-Jun.2020
  - Our first GPU System (Mar.2017), DDN IME (Burst Buffer)

- Oakforest-PACS (OFP) (Fujitsu, Intel Xeon Phi (KNL))
  - JCAHPC (U.Tsukuba & U.Tokyo)
  - 25 PF, #6 in 48th TOP 500 (Nov.2016) (#1 in Japan)
  - Omni-Path Architecture, DDN IME (Burst Buffer)

Code Name	KNL	BDW	FX10
Architecture	Intel Xeon Phi 7250 (Knights Landing)	Intel Xeon E5-2695 v4 (Broadwell-EP)	SPARC IX fx
Frequency (GHz)	1.40	2.10	1.848
Core # (Max Thread #)	68 (272)	18 (18)	16 (16)
Peak Performance (GFLOPS)	3,046.4	604.8	236.5
Memory (GB)	MCDRAM: 16 DDR4: 96	128	32
Memory Bandwidth(GB/sec., Stream Triad)	MCDRAM: 490 DDR4: 80.1	65.5	64.7
Out-of-Order	Y	Y	N
System	Oakforest-PACS	Reedbush-U	Oakleaf-FX

Code Name	KNL	BDW	FX10
Architecture	Intel Xeon Phi 7250 (Knights Landing)	Intel Xeon E5-2695 v4 (Broadwell-EP)	SPARC IX fx
Frequency (GHz)	1.40	2.10	1.848
Core # (Max Thread #)	68 (272)	18 (18)	16 (16)
Peak Performance (GFLOPS)/core	**44.8**	**33.6**	**14.8**
Memory Bandwidth(GB/sec., Stream Triad)/core	**MCDRAM: 7.21 DDR4: 1.24**	**3.64**	**4.04**
Out-of-Order	Y	Y	N
Network	**Omni-Path Architecture**	**Mellanox EDR Infiniband**	**Tofu 6D Torus**

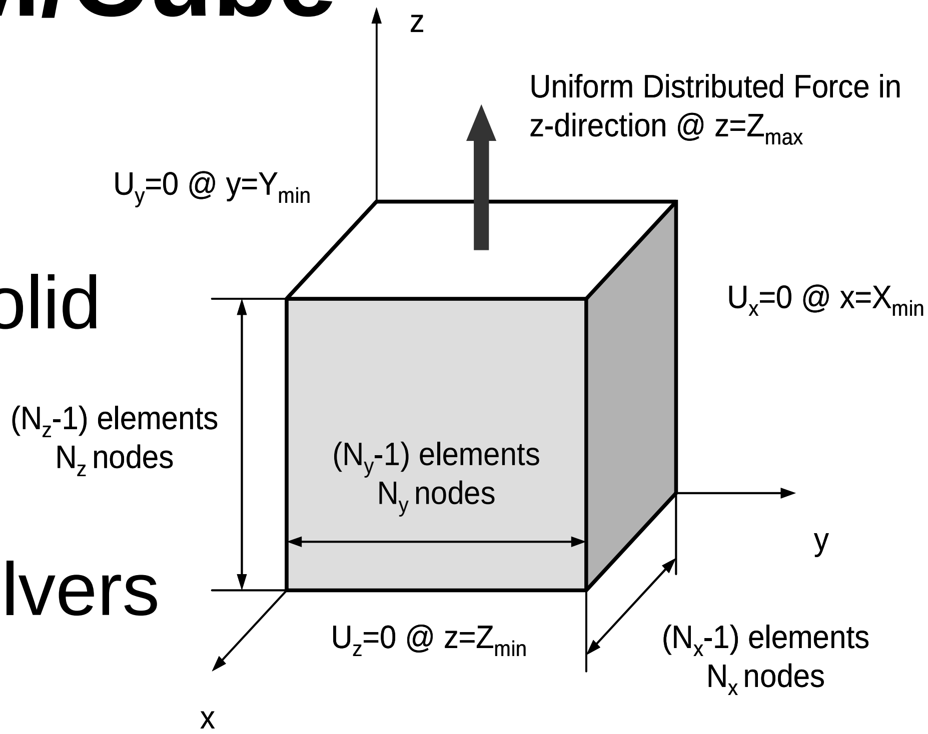
- Introduction
- Dynamic Loop Scheduling
- Hardware Environment

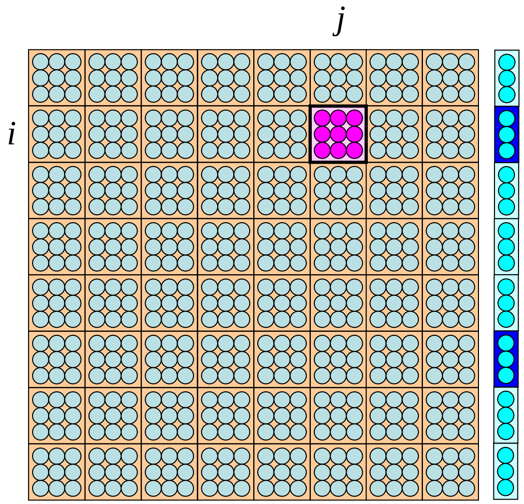
- **Preliminary Results by Parallel FEM**
- Strong Scaling
- SAI Preconditioning

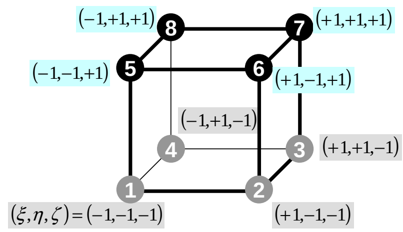
- Summary

GeoFEM/Cube

- Parallel FEM Code (& Benchmarks)
- 3D-Static-Elastic-Linear (Solid Mechanics)
- Performance of Parallel Preconditioned Iterative Solvers
  - 3D Tri-linear Hexahedral Elements
  - Block Diagonal LU + CG
  - Fortran90+MPI+OpenMP
  - Distributed Data Structure
  - MPI, OpenMP, OpenMP/MPI Hybrid
  - Block CRS Format







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

Configurations (1/2)

- Parallel Programming Model
  - Hybrid M×N (HB M×N)
    - "M": Number of OpenMP threads for each MPI process,
    - "N": Number of MPI processes on each CPU/socket.
  - FX10 and BDW: Flat MPI, HB 2×8, 4×4, 8×2, 16×1.
    - 16 of 18 cores on each socket used for BDW
  - Because each core of KNL can host up to four threads, we applied three configurations, 1T (1 thread per core), 2T (2 threads per core) and 4T (4 threads per core).
    - Therefore, M×N=64 for 1T, 128 for 2T, and 256 for 4T.
    - (1T) Flat MPI, HB 2×32, 4×16, 8×8, 16×4, 32×2, 64×1
    - (2T) HB 2×64, 4×32, 8×16, 16×8, 32×4, 64×2, 128×1
    - (4T) only HB 32×8 has been applied to limited cases.
    - **Flat/Quadrant, Only MCDRAM**
  - Each core of BDW can host two threads by hyper-threading, but this capability is deactivated on the Reedbush-U.

Configurations (2/2)

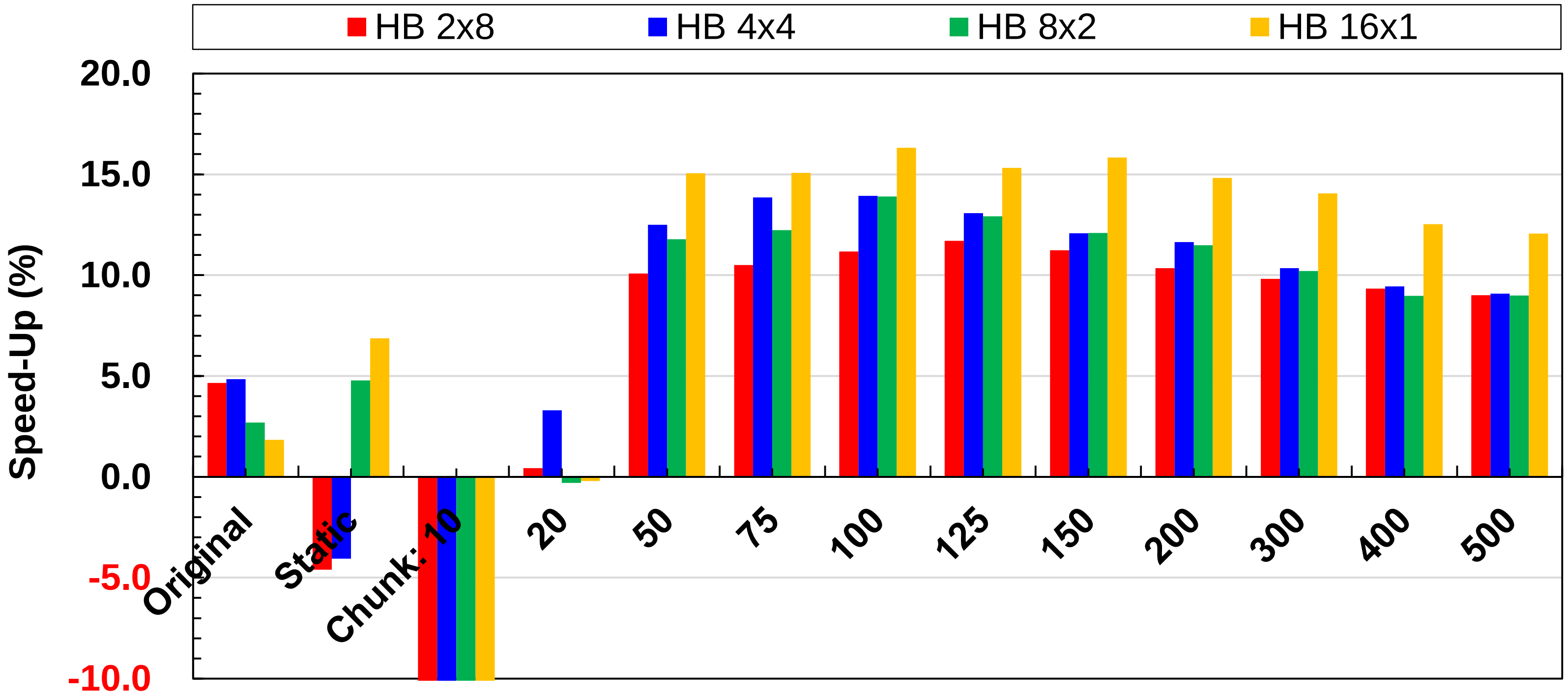
- **Original**
  - Original code without any *CC-Overlapping*. Local computation of SpMV starts after completion of HALO exchange.
- **Static**
  - *CC-Overlapping* with *static* loop scheduling is applied.
- **Dynamic**
  - *CC-Overlapping* with *dynamic* loop scheduling is applied.
  - *Chunk size* has been changed from 10 to 500.
- **Measurement: 5 times, Median's (and error-bar's) are shown**

Target Problem

- Performance of GeoFEM/Cube with 3,840 cores of each system
  - 240 nodes of FX10
  - 120 nodes (240 sockets) of BDW
  - 60 nodes of KNL
- The 1st problem includes 122,880,000 FEM nodes (=640×480×400), and 368,640,000 DOF
  - each CPU/socket of FX10 and BDW has 512,000 nodes (= 80×80×80), and 1,536,000 DOF.
- The 2nd problem includes 800×600×500 nodes, and 720,000,000 DOF
  - 100^3 nodes for each CPU/socket of FX10 and BDW

Preliminary Results: FX10

240 nodes, 3,840 cores, 368,640,000 DOF (=640×480×400×3),
Improvement of CG from Original Flat MPI



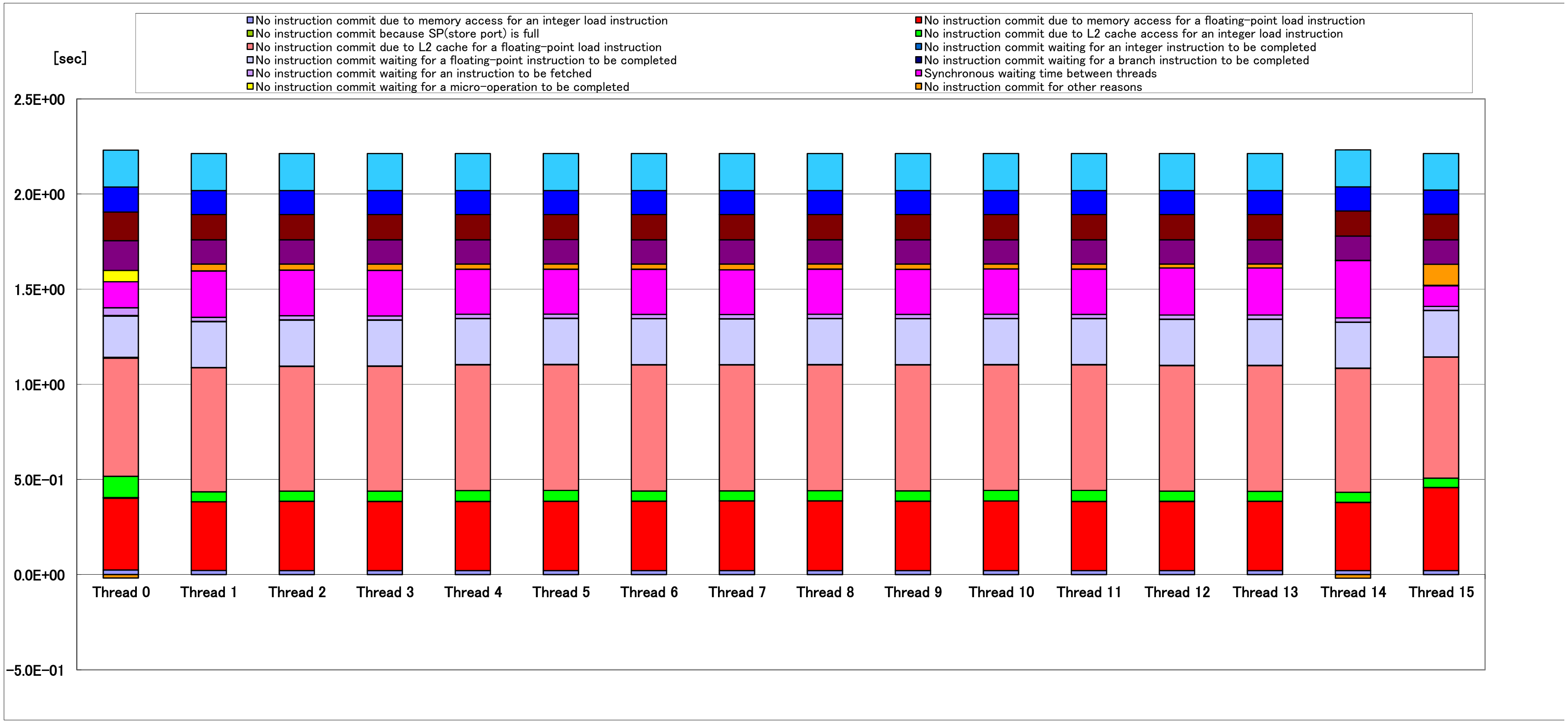
FX10: 240 nodes, 368,640,000 DOF

HB 16x1, Performance Analysis by Fujitsu's Profiler (single node)

	Original	Static	Dynamic: Chunk Size=100	Dynamic: Chunk Size=500
GFLOPS/node	12.59	13.33	14.47	13.82
Memory Throughput (GB/sec)	61.61	64.86	69.44	68.07
L2 Throughput (GB/sec)	71.13	75.03	84.15	79.03
sec./(100 iterations)	2.21	2.10	1.93	2.00
Synchronous waiting time between threads (sec) Averaged	.229	.122	.073	.061
L2 waiting for FP Load (sec) Averaged	.655	.657	.540	.614

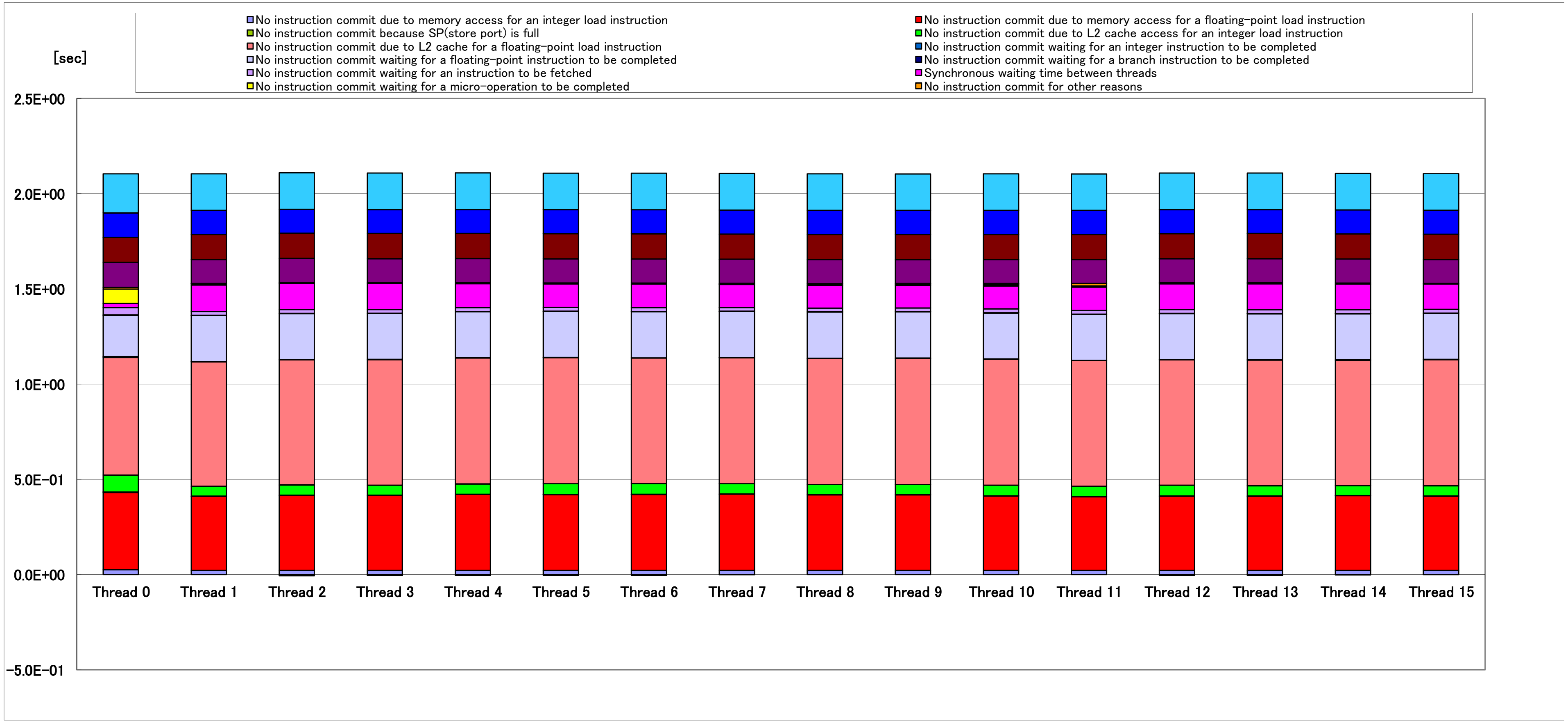
3,840 cores, PA Profiler
FX10: 240 nodes, 368,640,000 DOF
"Original": 2.21 sec.

■ Synchronization Waiting, ■ L2 Load



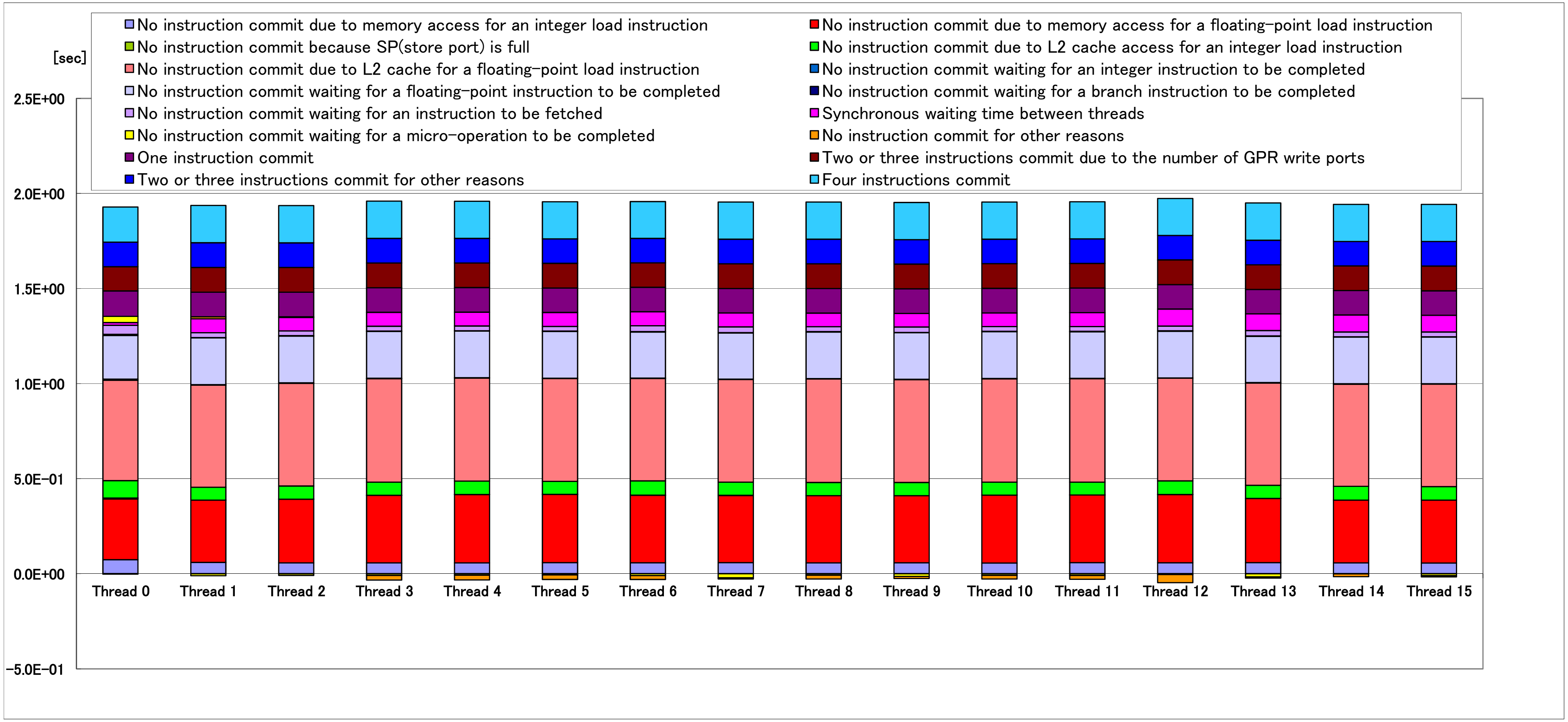
3,840 cores, PA Profiler FX10: 240 nodes, 368,640,000 DOF "Static": 2.10 sec.

■ Synchronization Waiting, ■ L2 Load



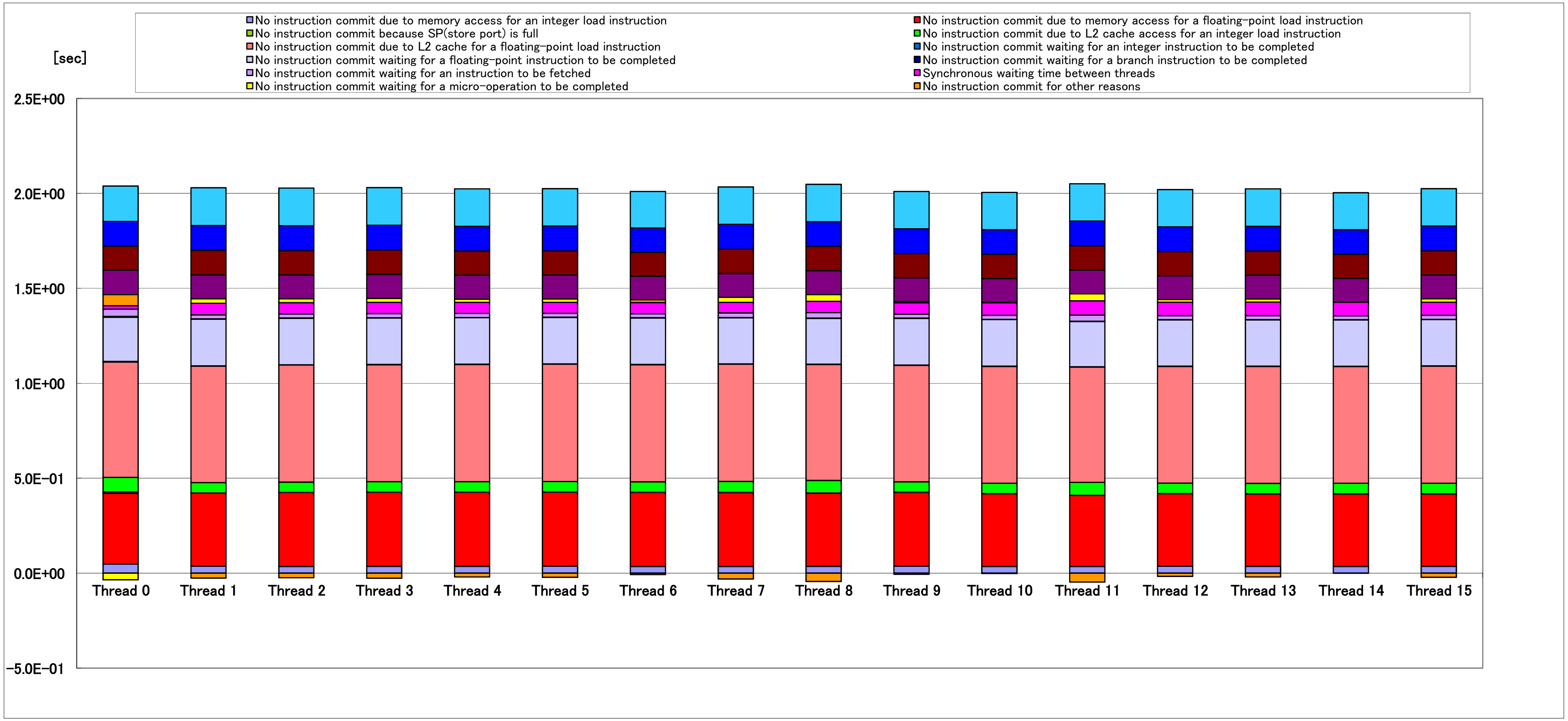
3,840 cores, PA Profiler FX10: 240 nodes, 368,640,000 DOF "Dynamic, Csz=100": 1.93 sec.

■ Synchronization Waiting, ■ L2 Load



3,840 cores, PA Profiler
FX10: 240 nodes, 368,640,000 DOF
“Dynamic, Csz=500”: 2.00 sec.

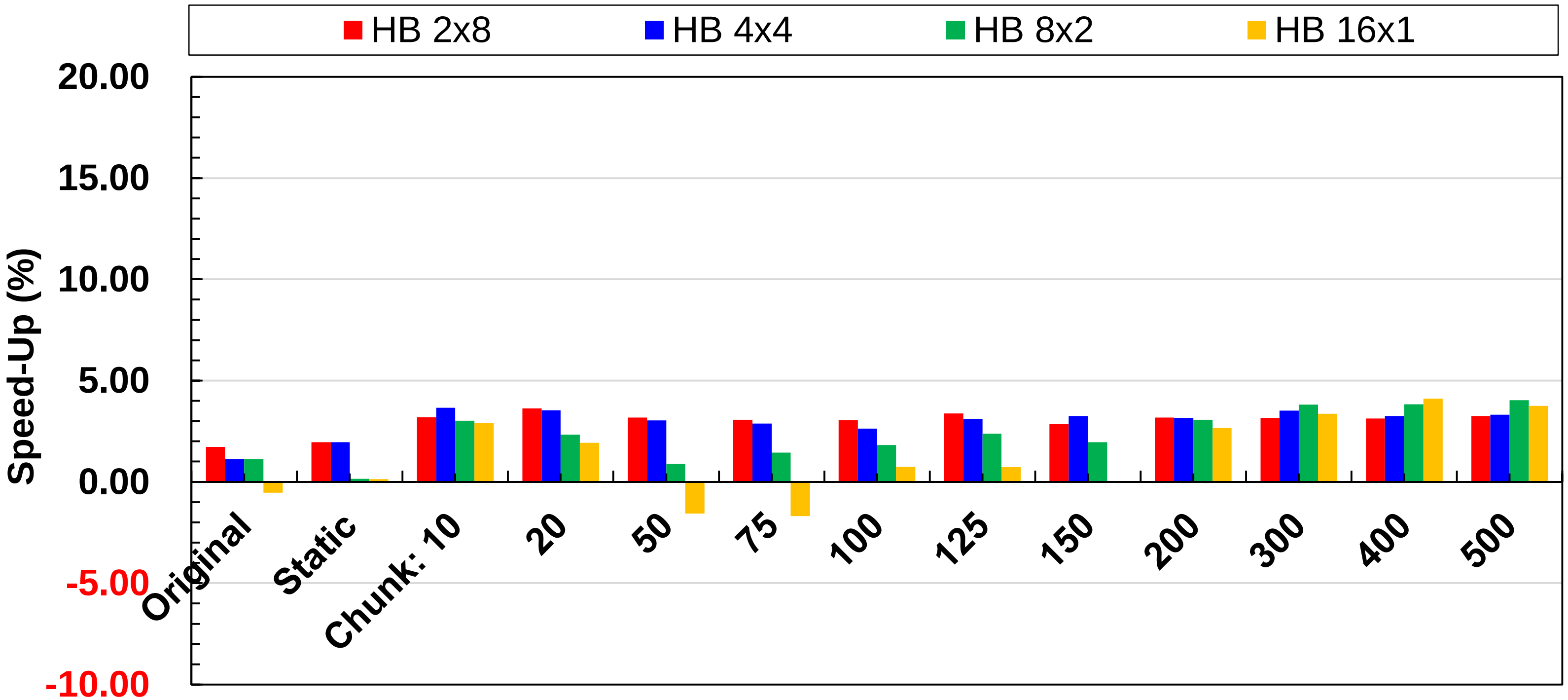
■ Synchronization Waiting, ■ L2 Load



Preliminary Results: BDW

120 nodes, 3,840 cores, 368,640,000 DOF (=640×480×400×3),

Improvement of CG from Original Flat MPI

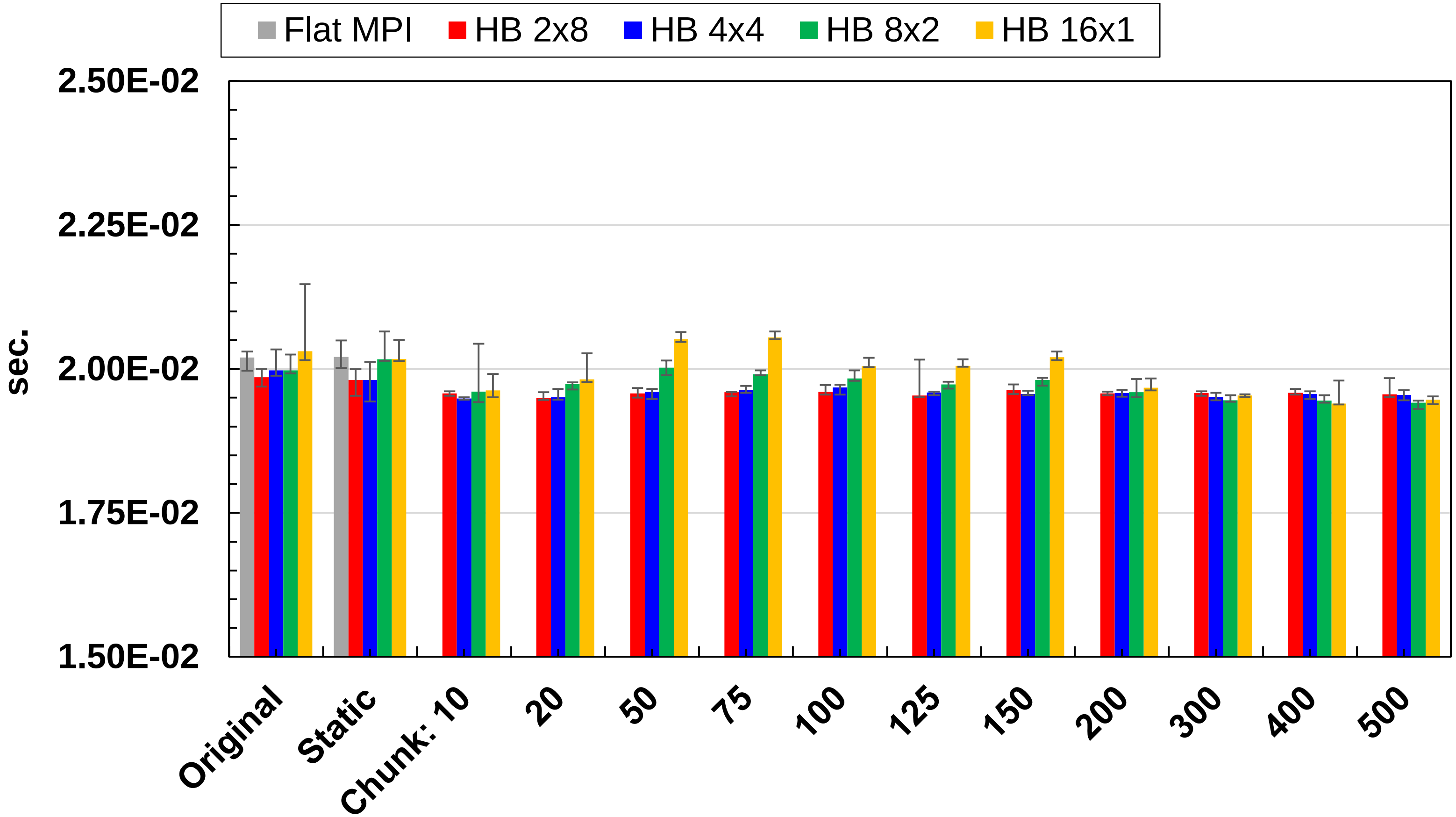


Preliminary Results: BDW

120 nodes, 3,840 cores, 368,640,000 DOF (=640×480×400×3),

Computation of Time of CG/Iteration

Error-bar shows max/min values of 5 measurements



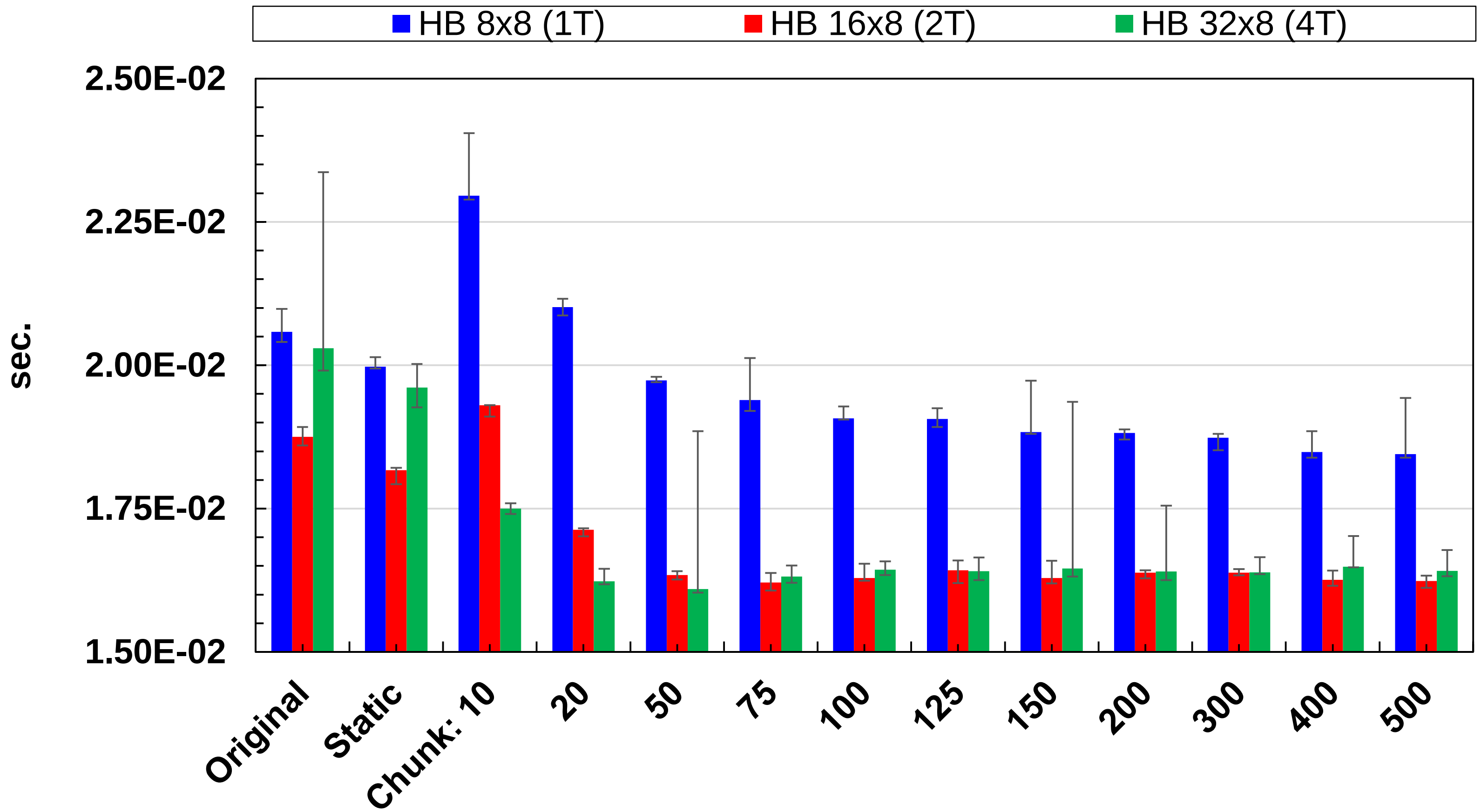
Preliminary Results: KNL

60 nodes, 3,840 cores, 368,640,000 DOF

Computation of Time of CG/Iteration

Error-bar shows max/min values of 5 measurements

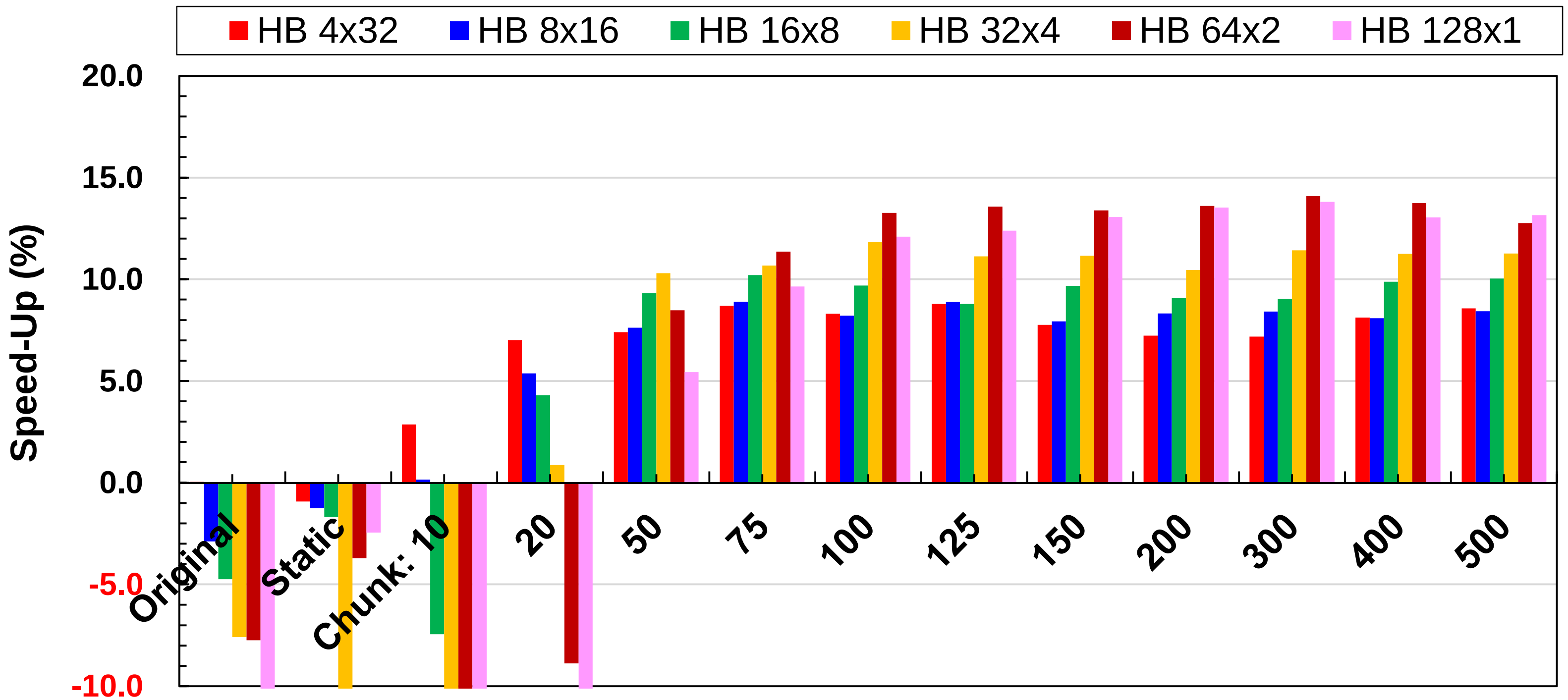
8 cores/MPI proc, Effects of Thread/Core (1T, 2T, 4T)



Preliminary Results: KNL/2T

60 nodes, 3,840 cores, 368,640,000 DOF

Improvement of CG from Original HB 2×64

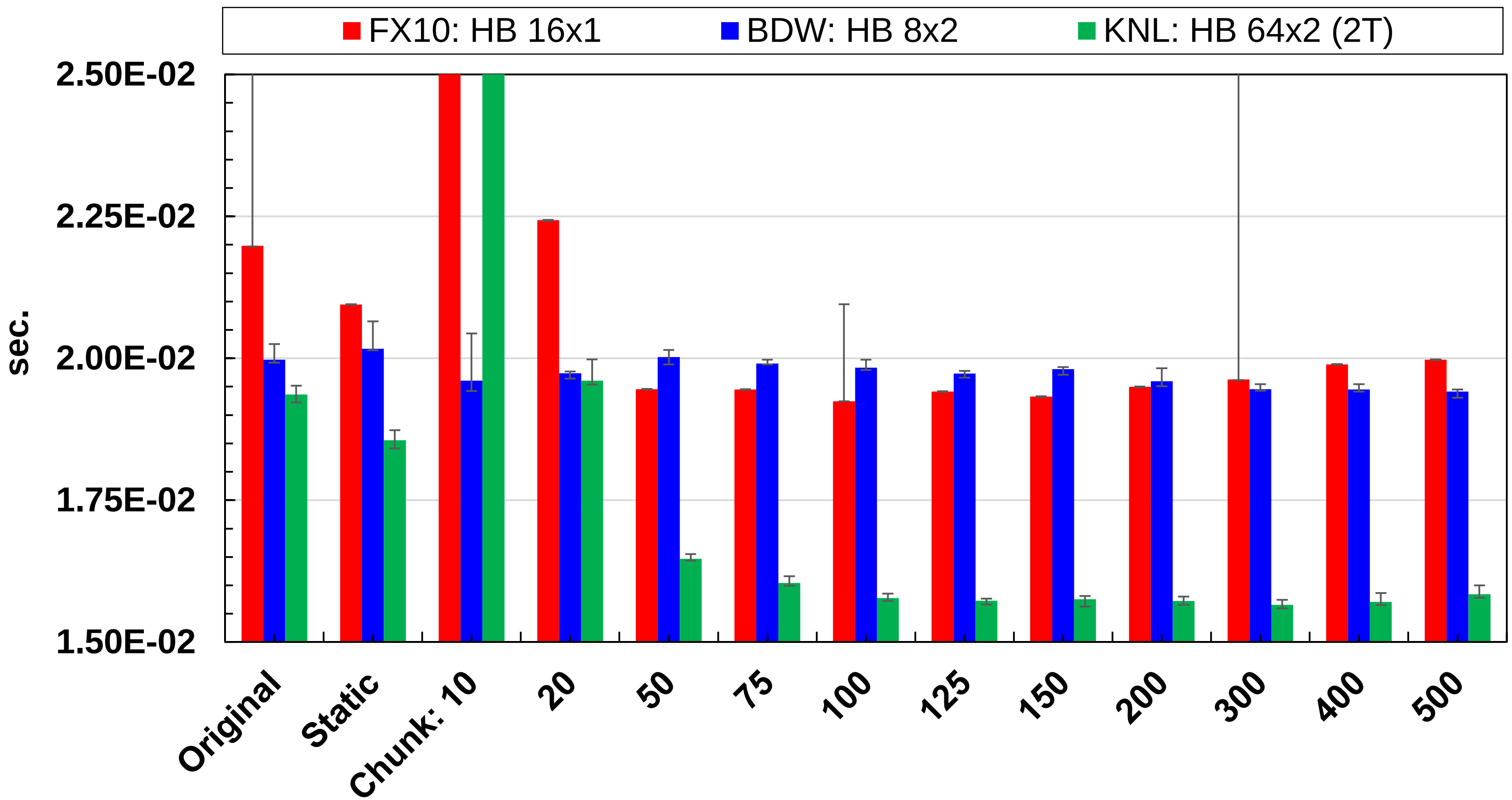


Preliminary Results: Best Cases

3,840 cores, 368,640,000 DOF

Computation of Time of CG/Iteration

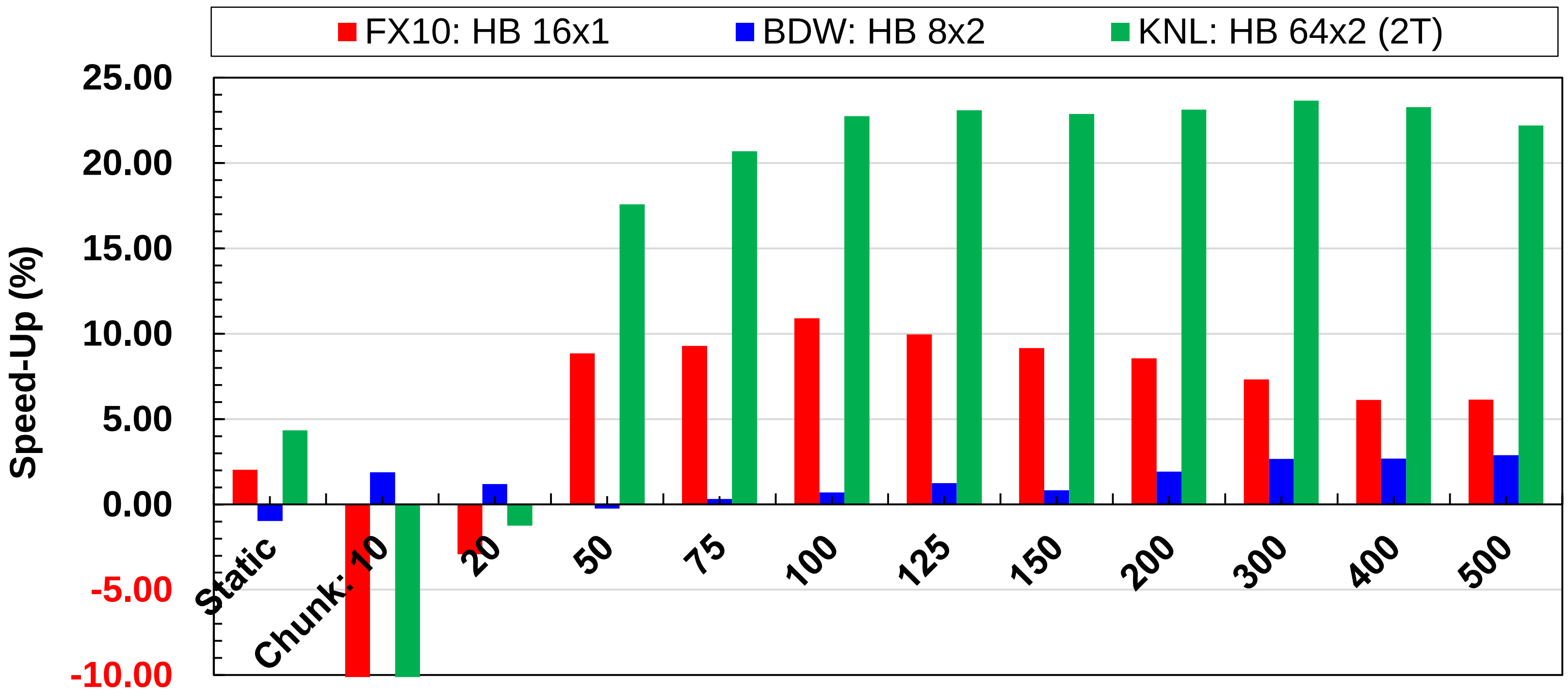
Error-bar shows max/min values of 5 measurements



Preliminary Results: Best Cases

3,840 cores, 368,640,000 DOF

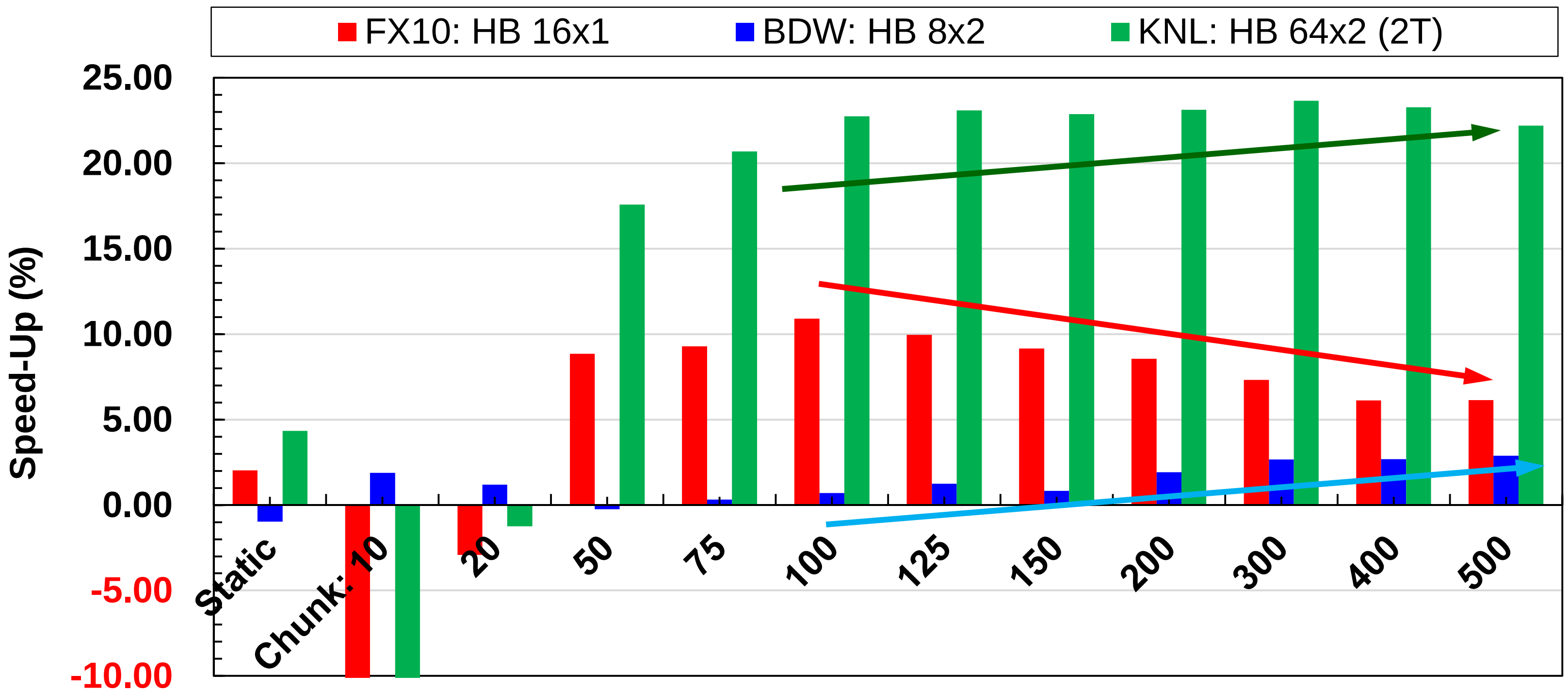
Improvement of CG from Original Cases



Preliminary Results: Best Cases

3,840 cores, 368,640,000 DOF

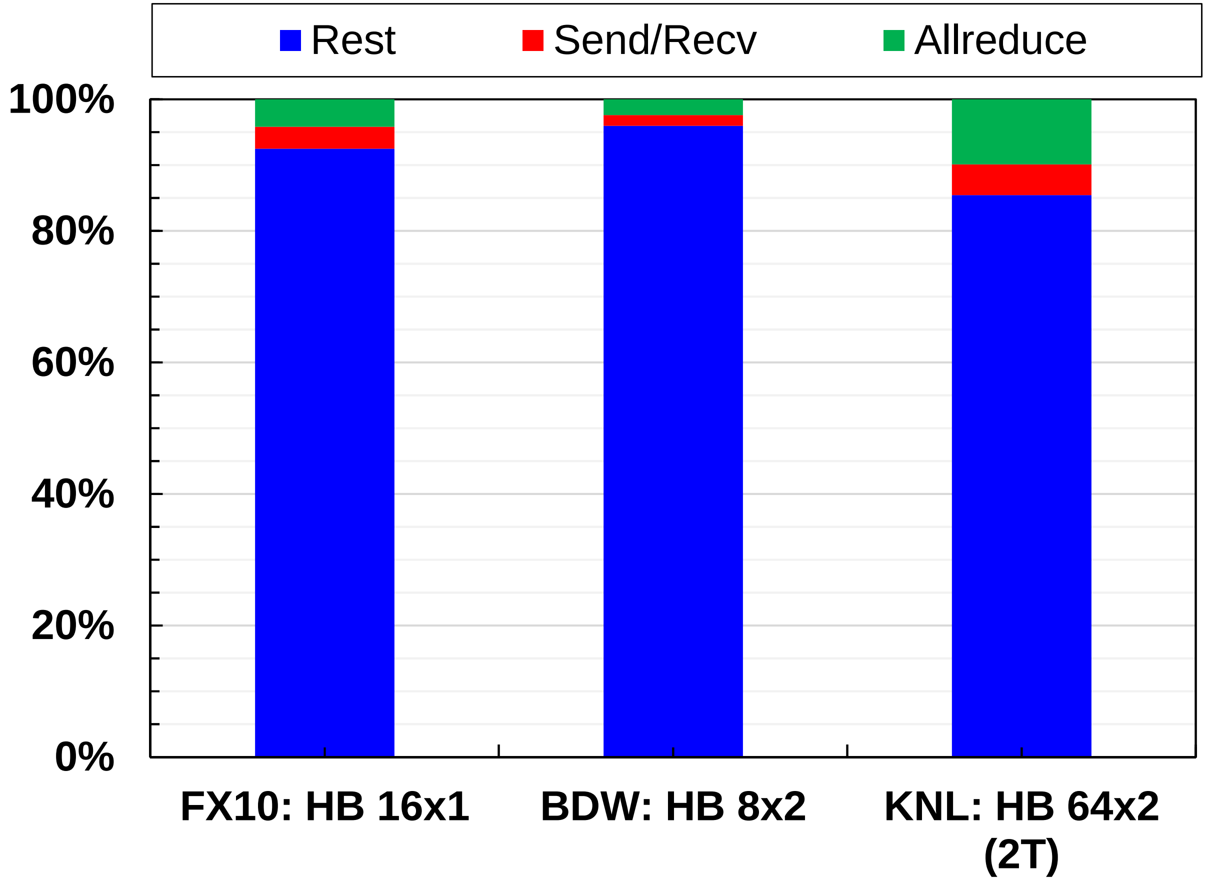
Improvement of CG from Original Cases



Preliminary Results: Original Cases

3,840 cores, 368,640,000 DOF

Communication Overhead by Collective/Point-to-Point Communications



7.5% 4.0% 14.6%

Features

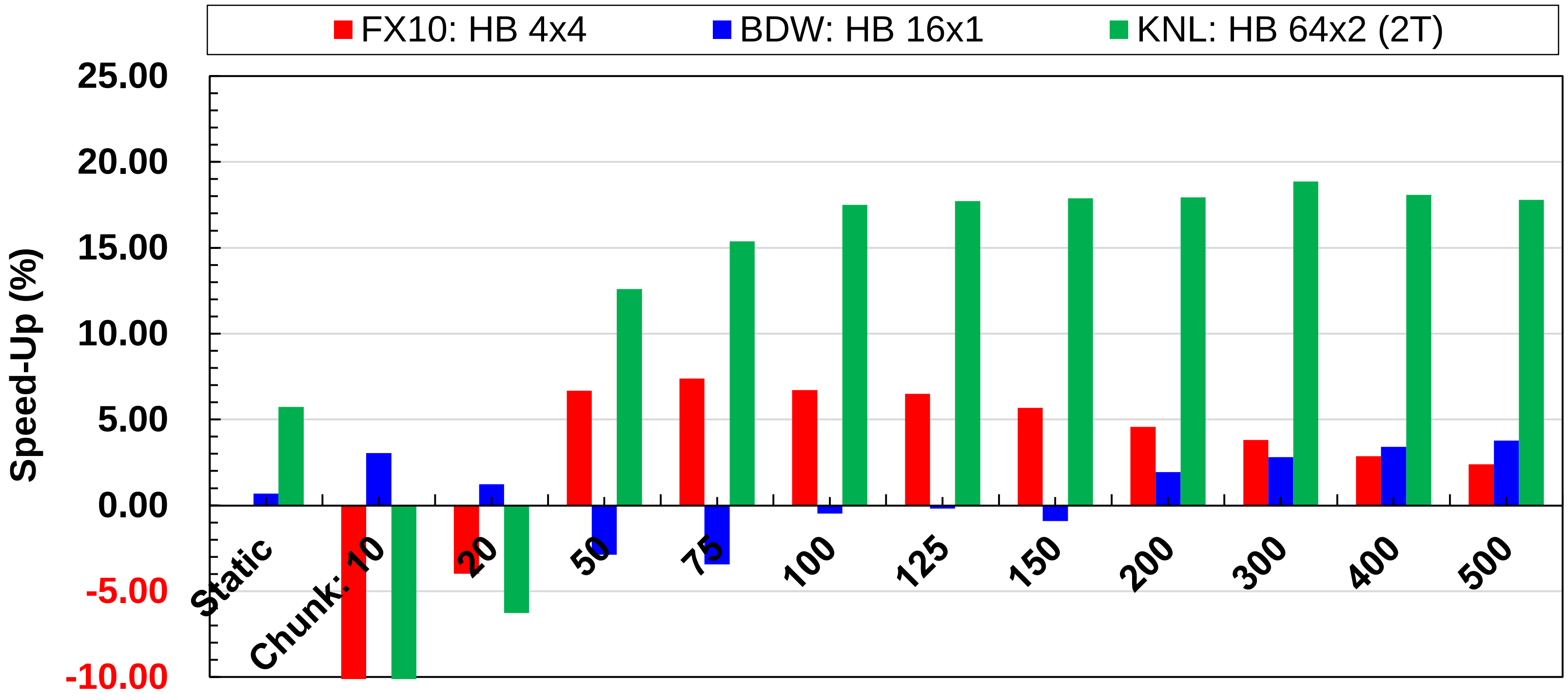
	Effect of Dynamic Scheduling	Optimum Chunk Size	Notes
FX10	Medium	100	Memory Throughput
BDW	Small	500+	Low Comm. Overhead Small number of threads.
KNL	Large	300-500	Effects are significant for HB 64x2, 128x1, where loss of performance by communications on master thread is rather smaller.

Preliminary Results: Best Cases

3,840 cores, **720,000,000 DOF**

Improvement of CG from Original Cases

Effects are Smaller (DOF/MPI Proc. is larger)



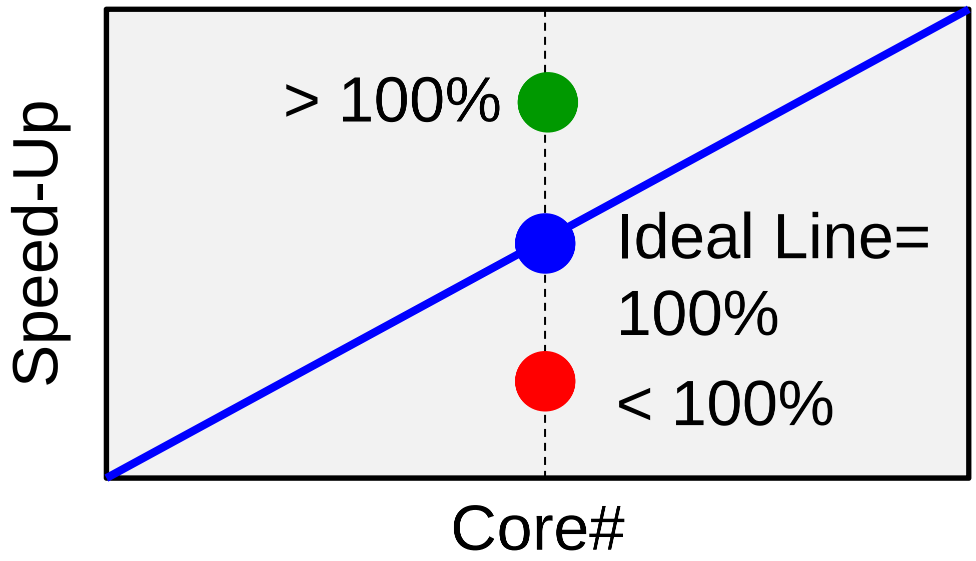
- Introduction
- Dynamic Loop Scheduling
- Hardware Environment

- Preliminary Results by Parallel FEM (GeoFEM/Cube)
- **Strong Scaling**
- SAI Preconditioning

- Summary

Target Problem

- 256^3 FEM Nodes, 50,331,648 DOF
- Strong Scaling
  - FX10: 2-1,024 nodes (32-16,384 cores)
  - BDW: 2-512 sockets, 1-256 nodes (32-8,192 cores)
    - Reedbush-U has only 420 nodes of BDW
  - KNL: 4-256 nodes (256-16,384 cores)
- Parallel Performance
  - 100%: on the ideal line
  - < 100%: BELOW
  - > 100%: ABOVE



Strong Scaling: KNL

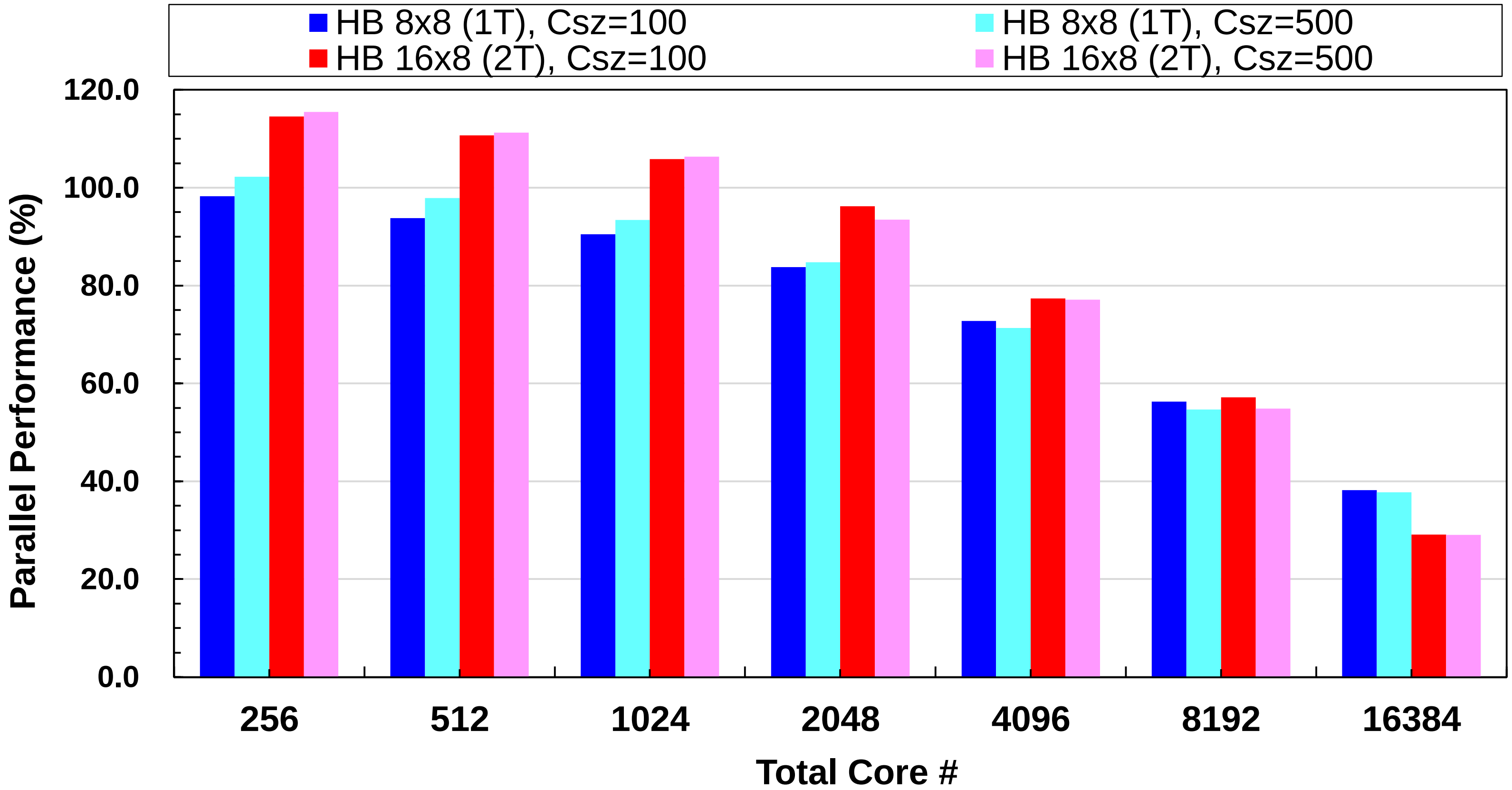
Parallel Performance (%)

50,331,648 DOF, 256-16,384 cores

Computation Time of Flat MPI at 256 cores: 100%

HB 8×8 (1T) and HB 16×8 (2T)

1T is better, if Core# increases



Strong Scaling

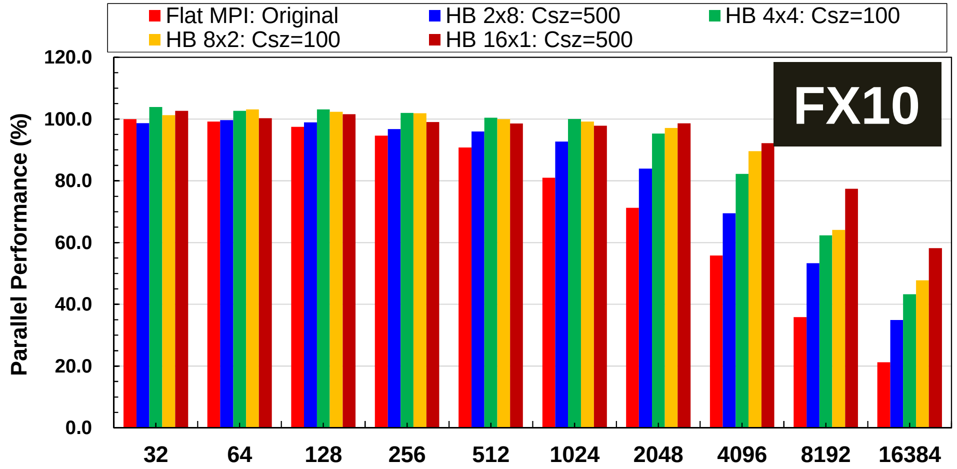
Parallel Performance (%)

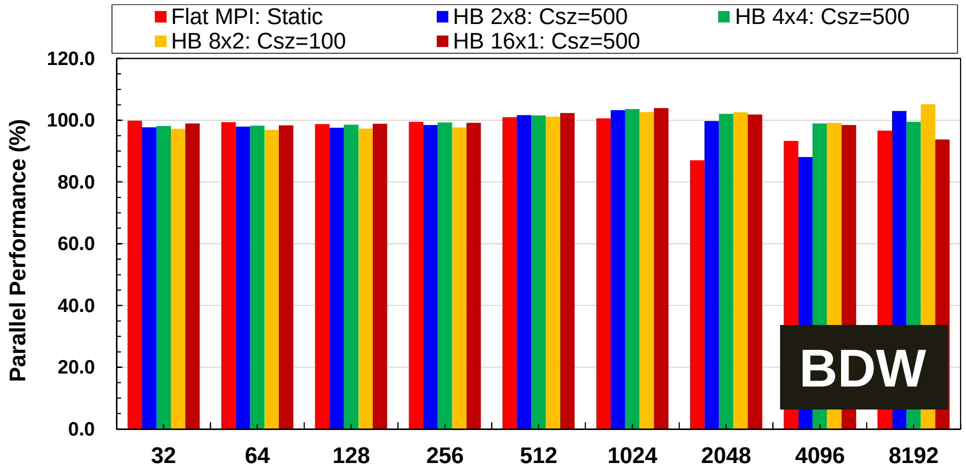
BEST case for each HB MxN

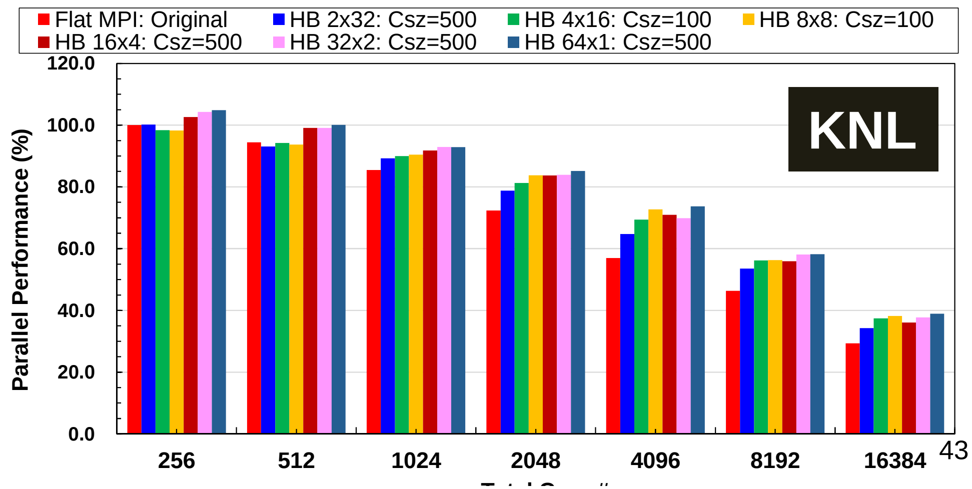
50,331,648 DOF

Computation Time of Flat MPI at Min.# cores: 100%

This difference between BDW and KNL might be because difference of performance between Infiniband EDR and Omni-Path Architecture.







Strong Scaling

Parallel Performance (%)

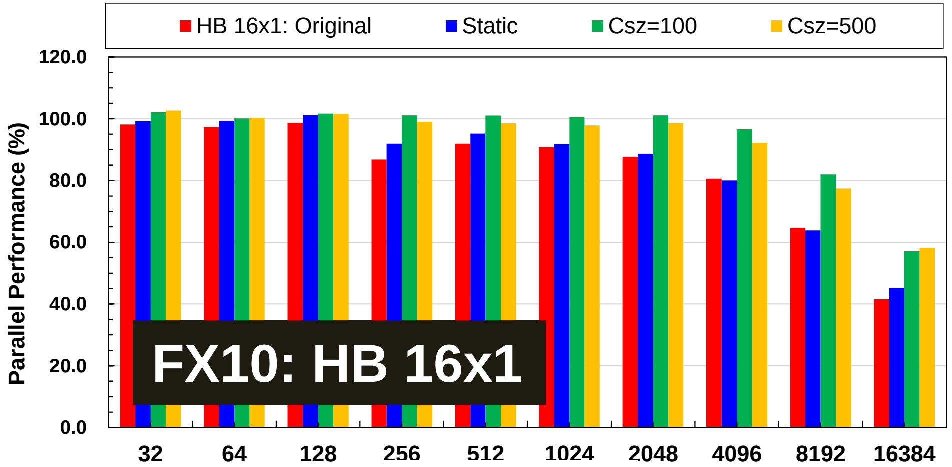
Effect of Dynamic Loop Scheduling

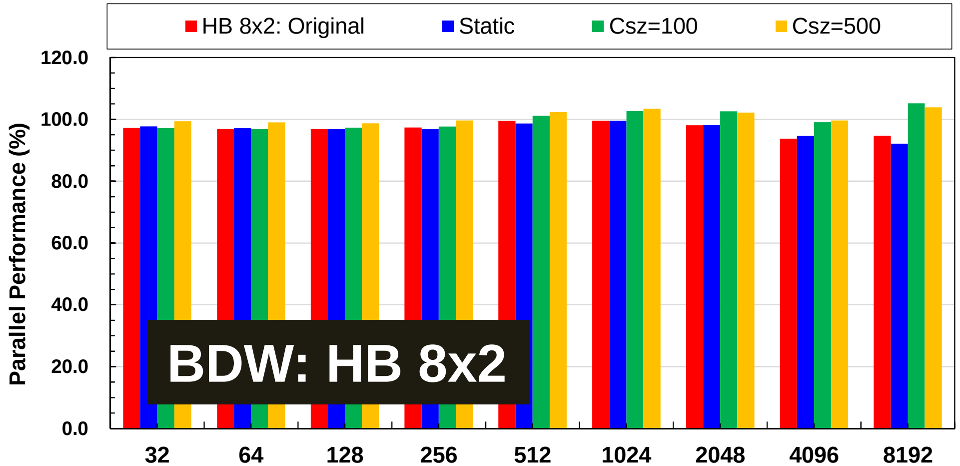
50,331,648 DOF

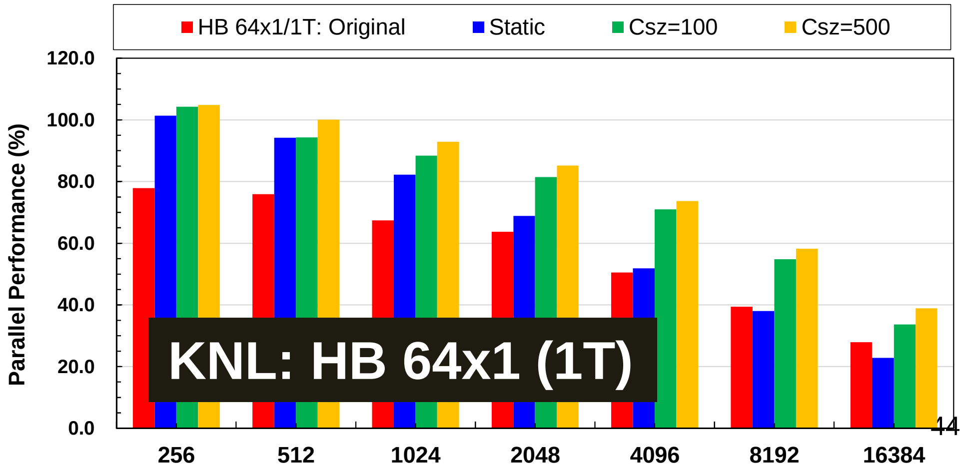
Computation Time of Flat MPI at Min.# cores: 100%

Effect of Dynamic Loop Scheduling with more than 8,192 cores

- FX10: 20%-40%
- BDW: 6%-10%
- KNL with HB 8×8 (1T): 20%-30%
- KNL with HB 64×1 (1T): 40%-50%







- Introduction
- Dynamic Loop Scheduling
- Hardware Environment

- Preliminary Results by Parallel FEM (GeoFEM/Cube)
- Strong Scaling
- **SAI Preconditioning (Sparse Approximate Inverse)**

- Summary

Next Target

- SAI (Sparse Approximate Inverse)
  - Mat-Vec. Multiplication for Sparse Approximate Inverse Matrix
  - Much Easier than ILU (failed)
  - Old, but not so bad. Suitable for GPU.
  - SAI: Various Approaches: SPAI (Explicit SAI) adopted

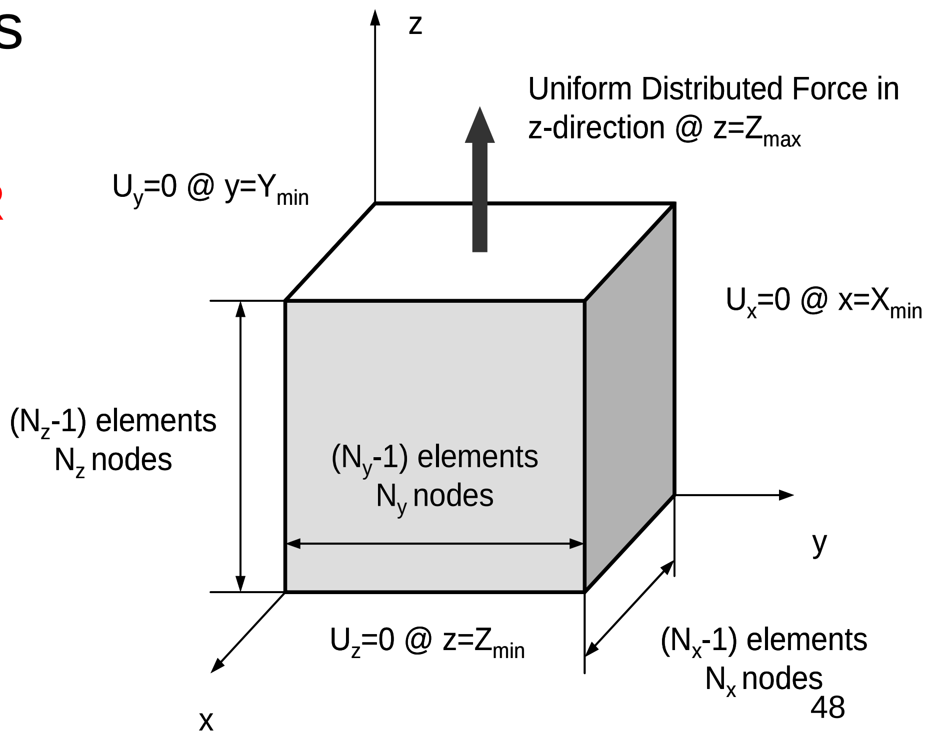
SAI (Sparse Approx. Inverse)

- Preconditioning method for sparse matrices derived from localized-type scientific applications, such as FEM, FDM, FVM etc.
- **Define inverse (preconditioned) matrix *[M]* explicitly.**
- Even if original matrix *[A]* is sparse, inverse is usually dense due to ***fill-in***.
- **Sparse approximate inverse (SAI)** is an *approximate* inverse matrix, which has as similar sparsity as the original matrix has.

$$sparsity(M) \approx sparsity(A)$$

GeoFEM-SAI/Cube

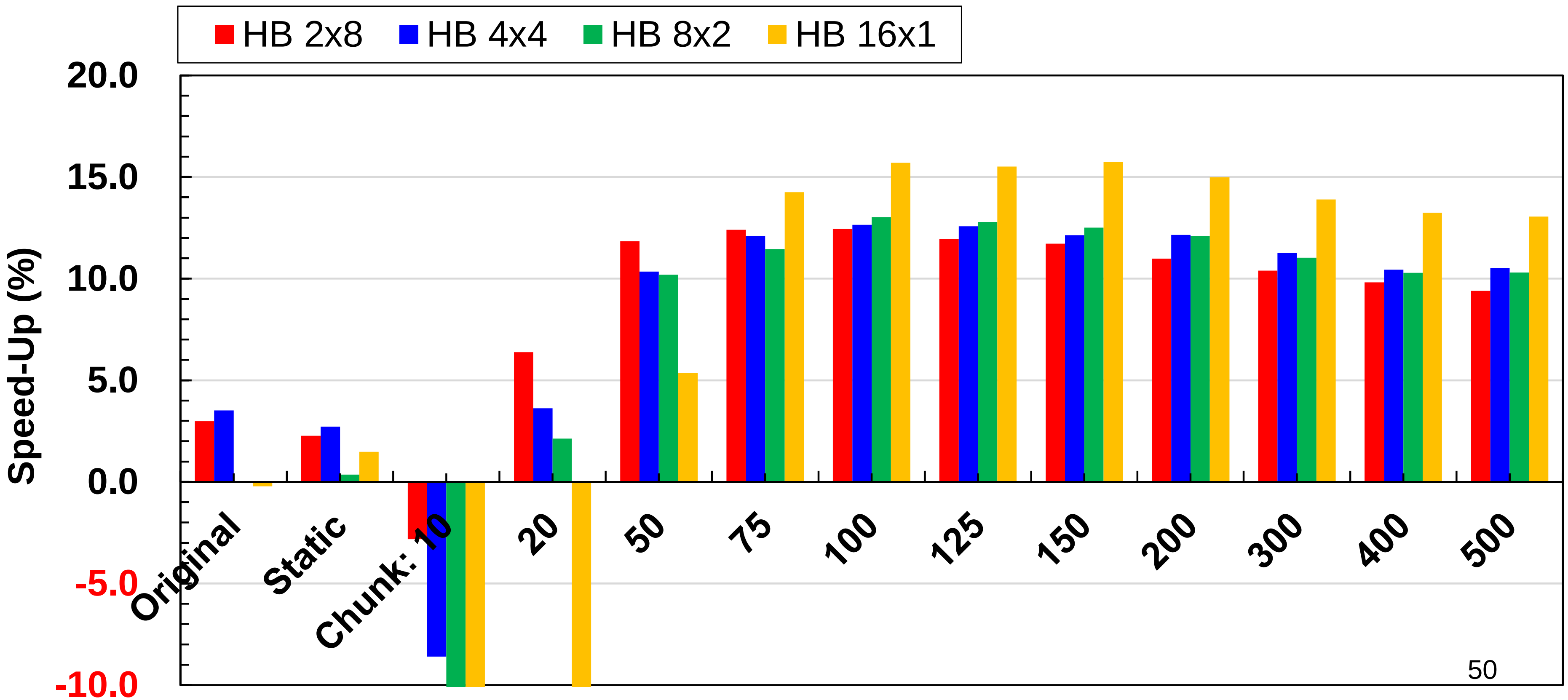
- Parallel FEM Code (& Benchmarks)
- 3D-Static-Elastic-Linear (Solid Mechanics)
- Performance of Parallel Preconditioned Iterative Solvers
  - 3D Tri-linear Hex. Elements
  - SAI + BiCGSTAB
    - Dropping Tolerance after QR Factorization: t
  - Fortran90+MPI+OpenMP
  - Distributed Data Structure
  - MPI, OpenMP, OpenMP/MPI Hybrid
  - Block CRS Format



Target Problem

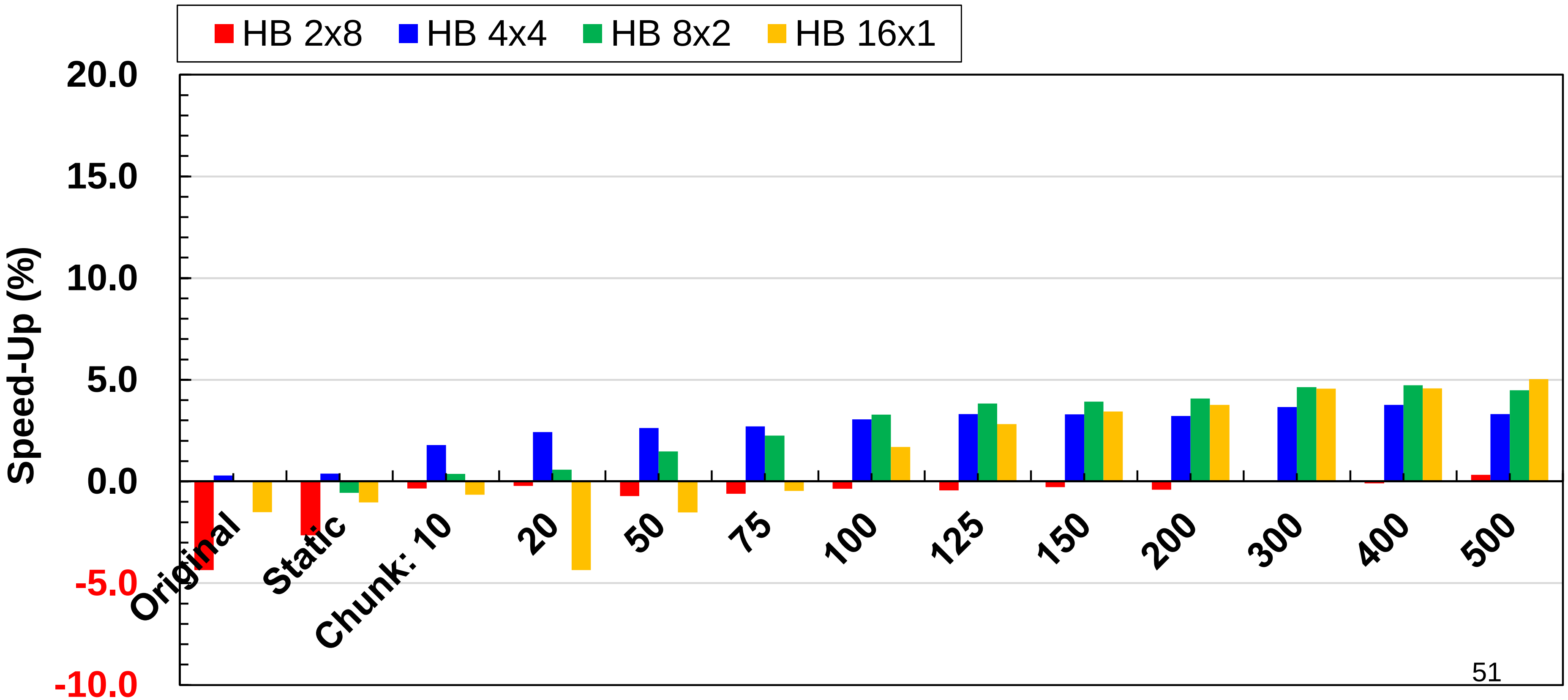
- 393,216,000 FEM nodes (=960×640×640), 1,179,648,000 DOF
  - each node of FX10 has 512,000 nodes (=80×80×80), and 1,536,000 DOF
  - Dropping tolerance *e* is set to 0.10.
  - Number of non-zero components of *M* is 25.8% of that of original *A*.
- Using 12,288 cores
  - FX10: 768 nodes
  - BDW: 768 sockets, 384 nodes
  - KNL: 192 nodes, 2 threads per core (2T)

FX10: 1,179,648,000 DOF
768 nodes, 12,288 cores
Speed-Up compared to “Original” HB 8x2, t=0.10



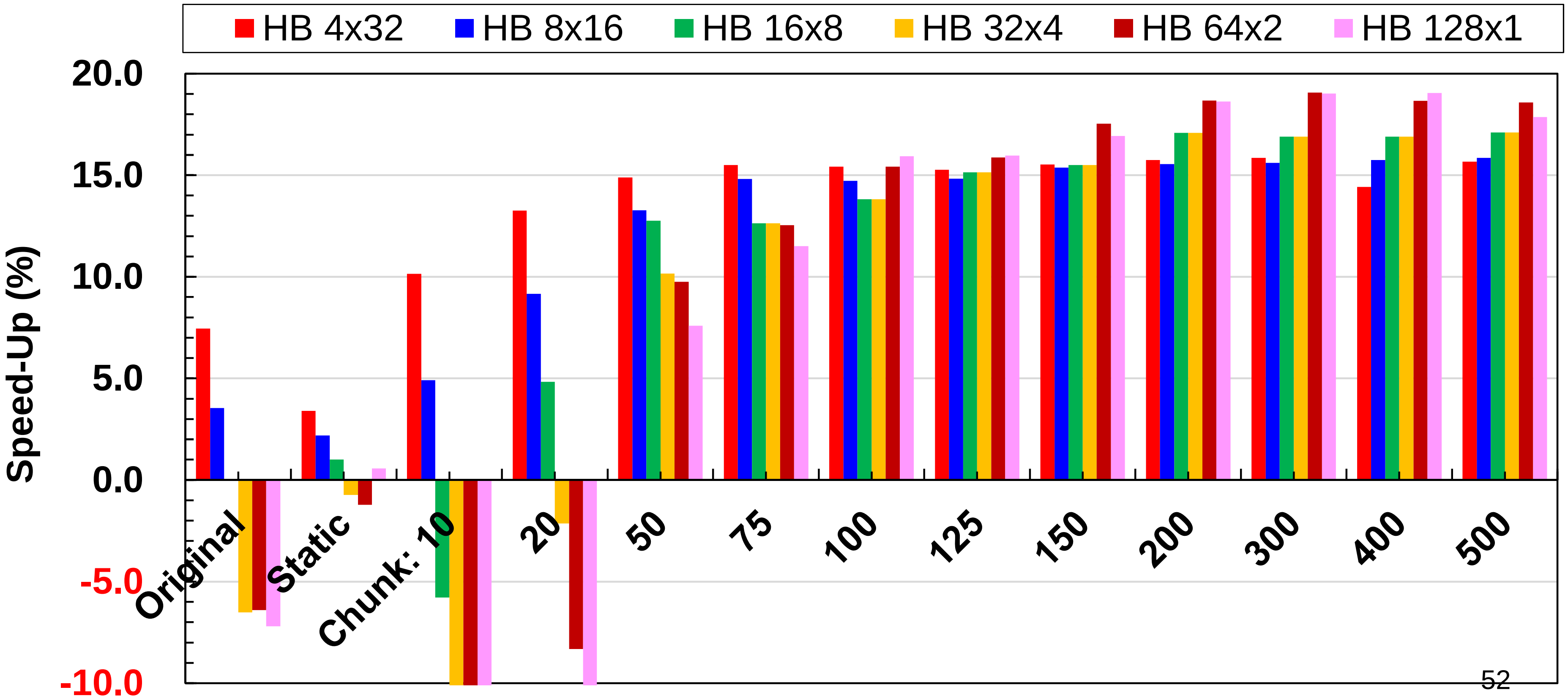
Reedbush-U (BDW): 1,179,648,000 DOF 384 nodes, 12,288 cores

Speed-Up compared to “Original” HB 8x2, t=0.10



Oakforest-PACS (KNL): 1,179,648,000 DOF 192 nodes, 12,288 cores, 2T/core

Speed-Up compared to “Original” HB 16x8, t=0.10



- Introduction
- Dynamic Loop Scheduling
- Hardware Environment

- Preliminary Results by Parallel FEM (GeoFEM/Cube)
- Strong Scaling
- SAI Preconditioning

- **Summary**

Summary (1/2)

- CC-Overlapping by Dyn. Loop Scheduling of OpenMP
- SpMV of CG/BiCGSTAB in Parallel FEM (GeoFEM)
  - Significant Effects by Dynamic Loop Scheduling
  - Improvement of Performance by 40%-50% in strong scaling using up to 16,384 cores of FX10 and KNL Clusters.
  - On the contrast, improvement of performance is very small on Intel Broadwell (BDW) cluster.
  - Generally, effect of CC-Overlapping with dynamic loop scheduling is significant, if thread number for each MPI process is larger.
  - Therefore, developed method is expected to be useful for manycore architectures with $O(10^2)$ cores, such as Intel Xeon Phi.

Summary (2/2)

- SAI-BiCGSTAB
  - 15%-20% improvement of performance has been obtained on 12,288 cores of Fujitsu FX10 and KNL cluster.
- CC-Overlapping with dynamic loop scheduling improves the performance of parallel iterative solvers significantly, although algorithm is very simple.
- Future Work
  - More complicated preconditioning method, such as ILU, MG
  - Combination with Pipelined Method
  - Automatic selection of optimum Chunk Size
  - Further Optimization: Strong Scaling on KNL Cluster
  - The developed method might not work on NUMA
    - Appropriate runtime software will be needed.