Multi-block/multi-core SSOR preconditioner for the QCD quark solver for the K computer

Ken-Ichi Ishikawa (Hiroshima Univ.)

In collaboration with

T.Boku, Y.Kuramasi, K.Minami, Y.Nakamura, F.Shoji, D.Takahashi, M.Terai, A.Ukawa, T.Yoshie

(RIKEN-Tsukuba Joint Research)

Plan of my talk

1. K computer
2. Lattice QCD on K computer
3. Lüscher's SAP preconditioner
4. Choice of the Block solver in a domain
5. Results
6. Summary

1. K computer

- Japanese national project
  - Developed By RIKEN and Fujitsu since 2006
  - Will be provided for public via HPCI in Sep. 2012.
- Over 10PFlops sustained speed in the LINPACK benchmark (TOP500 @ Nov. 2011)
- Over 80,000 nodes
  - Single CPU SPARC64 VIIIfx (8core@2GHz) / node
    - (SIMD fused multiply) x (2exec)x(8core) / cycle = 128GFlops(D.P.)
    - 256 FP registers/core, 6MB shared L2 cache/chip, hardware barrier.
  - 6D Mesh/Torus network connected by “Tofu (Torus fusion)” interconenct.
    - Can involve lower (1,2,3) dimensional torus networks without network reconnection.
    - Free from single point network failure
    - 3D torus x 3D mesh: 3D torus part is used to construct large 3D torus.
- Language : C,C++,Fortran
- Parallelization: MPI, OpenMP, 3D torus network for users.

2. Lattice QCD on K computer

- Lattice QCD is one of the suitable application for massively parallel supercomputer.
- In the RIKEN-Tsukuba joint research program, we have developed the lattice QCD program based on the Lüscher's Domain-decomposed HMC (DDHMC) algorithm (Clover quarks).
- In this talk we present some results from the performance tuning.

3. Lüscher's SAP preconditioner

- Our target algorithm is the DDHMC algorithm (Clover quarks).
- We tune and optimize the quark solver part.
- The quark solver uses
  - Nested BiCGStab algorithm (Sakurai-Tadano)
    - Outer solver : BiCGStab Double precision (with flexible preconitioner)
    - Inner solver : BiCGStab Single precision (as a preconditioner for the outer)
      - The inner solver is further perconsitioned by the Lüscher's SAP preconditioner.
- The inner solver consumes the most of computational time in the DDHMC algorithm.
- We tune and optimize the Single precision inner solver performance.

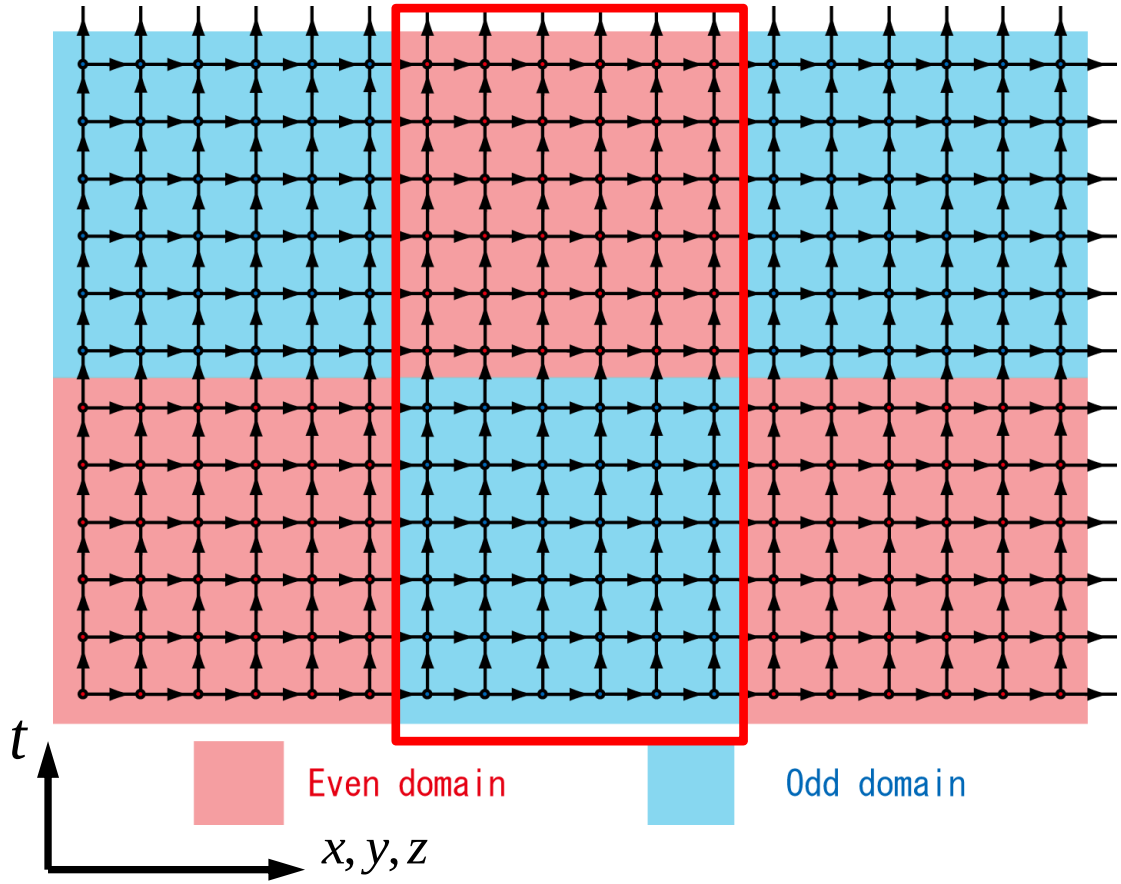
3. Lüscher's SAP preconditioner

- Based on Even-Odd domain decomposition

$$Dx = b$$

$$D = \begin{pmatrix} D_{EE} & D_{EO} \\ D_{OE} & D_{OO} \end{pmatrix}$$

- 2 Domain blocks in a node.
- 6^4 size for a block.



D: Clover term preconditioned Wilison-Clover Dirac op.

D_{EE}, D_{OO}: Even domain (Odd domain) restricted

D_{EO}, D_{OE}: Odd to Even domain (Even to Odd domain) connecting

- SAP preconditioner $M_{SAP} \approx D^{-1}$
  - A Neuiman iteration for 1/D.

$$M_{SAP} = K \sum_{j=0}^{N_{SAP}-1} (1-DK)^j \xrightarrow{N_{SAP}\to\infty} K(DK)^{-1} = D^{-1}$$

when $|DK|<1$.

  - Matrix : K
    - DK has a small condition number than that of D.

$$K = \begin{pmatrix} D_{EE}^{-1} & 0 \\ -D_{OO}^{-1} D_{OE} D_{EO}^{-1} & D_{OO}^{-1} \end{pmatrix}$$

$$DK = \begin{pmatrix} D_{EE} & D_{EO} \\ D_{OE} & D_{OO} \end{pmatrix} \begin{pmatrix} D_{EE}^{-1} & 0 \\ -D_{OO}^{-1} D_{OE} D_{EO}^{-1} & D_{OO}^{-1} \end{pmatrix} = \begin{pmatrix} 1 - D_{EO} D_{OO}^{-1} D_{OE} D_{EO}^{-1} & D_{EO} D_{OO}^{-1} \\ 0 & 1 \end{pmatrix}$$

- The linear equation is preconditioned as

$$Dx = b \Rightarrow (DM_{SAP})z = b, x = M_{SAP} z$$

This is solved with the Single precision BICGStab

4. Choice of the Block solver in a domain

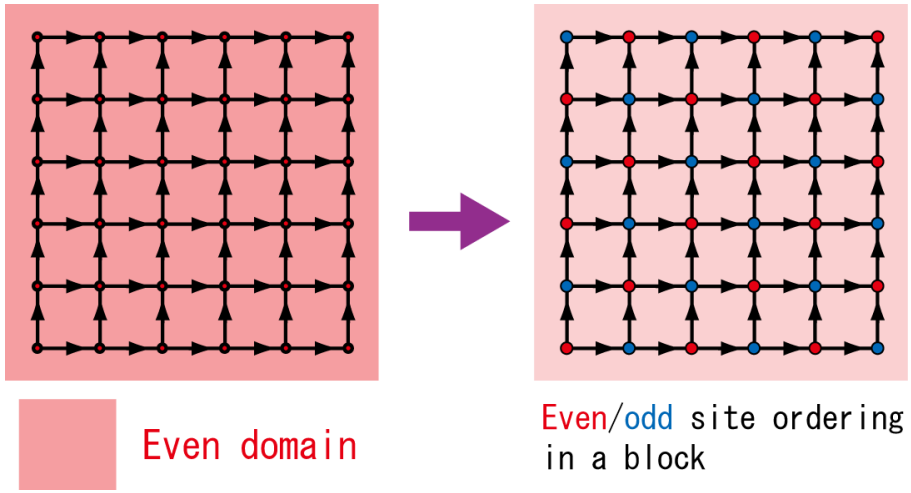
- The computational and algorithmic performance of the SAP preconditioner depends on:
  - Kernel routines : $D_{EE}, D_{OO}, D_{EO}, D_{OE}$
  - Block inversion in a domain : $(D_{EE})^{-1}, \ (D_{OO})^{-1}$
- Approximate solution for $(D_{EE})^{-1}, (D_{OO})^{-1}$ is sufficient for SAP.
  - Stationary fixed iterative method (MR, Neuman..) with:
    - Even/odd site preconditioning in a block (Lüscher)
    - SSOR with natural ordering in a block (PACS-CS) [for Single thread]

$$A_{EE} \approx (D_{EE})^{-1}, \quad A_{OO} \approx (D_{OO})^{-1}$$

- The kernel routines are optimized to make use of the full functionality of the SIMD and the many registers.
- We have to use 8-cores to achieve high efficiency for the K computer. We extend the SSOR solver to work with the 8-core threading.

- OpenMP parallelization of the block solver.
  - Even/odd site preconditioning : easy to extend to OpenMP 8 threading.

$$D_{EE} = \begin{pmatrix} (1_{EE})_{ee} & -\kappa(M_{EE})_{eo} \\ -\kappa(M_{EE})_{eo} & (1_{EE})_{oo} \end{pmatrix}$$



$$(D_{EE})^{-1} v_E \approx \begin{pmatrix} (z_E)_e \\ (v_E)_o + \kappa(M_{EE})_{oe}(z_E)_e \end{pmatrix}$$

$$(z_E)_e = \sum_{j=0}^{Nblk-1} \left[(1_{EE})_{ee} - (\hat{D}_{EE})_{ee}\right]^{-1} \left((v_E)_e + \kappa(M_{EE})_{eo}(v_E)_o\right)$$

$$(\hat{D}_{EE})_{ee} = (1_{EE})_{ee} - \kappa^2 (M_{EE})_{eo}(M_{EE})_{oe}$$

All even sites are independent. The site loop can be OpenMP parallerized. (put Open MP directives)

- OpenMP parallelization of the block solver.
  - SSOR natural ordering preconditioning : Recursive dependency in the forward and backward substitutions when a simple natural ordering is applied in the block.
    - SSOR in LQCD [S.Fischer, A.Frommer, U.Glässner, Th.Lippert, G.Ritzenhöfer, K.Schilling, CPC 98 (1996), Th.Lippert, P.Comp. 25 (1999)]

$$D_{EE} = 1_{EE} + L_{EE} + U_{EE}$$

$$L_{EE}: \text{strictly lower triangular part},\ U_{EE}: \text{strictly upper triangular part}$$

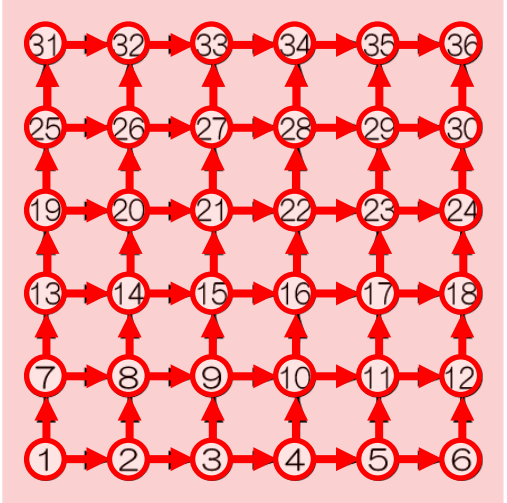
$$(D_{EE})^{-1} v_E \approx (1_{EE} + \omega L_{EE})^{-1} \sum_{j=0}^{Nblk-1} \left[1_{EE} - \left(\hat{D}_{EE}\right)_{SSOR}\right]^j (1_{EE} + \omega U_{EE})^{-1}$$

$$\left(\hat{D}_{EE}\right)_{SSOR} = \frac{1}{\omega}\left\{(1_{EE} + \omega L_{EE})^{-1}\left[1_{EE} + (2-\omega)(1_{EE} + \omega U_{EE})^{-1}\right] + (1_{EE} + \omega U_{EE})^{-1}\right\}$$

  - The Structure of L_{EE}, U_{EE} depends on the site indexing in a block.
  - The inversions $(1_{EE} + \omega L_{EE})^{-1}, (1_{EE} + \omega U_{EE})^{-1}$ are solved with forward or backward substitution.
  - The computational cost of $\left(\hat{D}_{EE}\right)_{SSOR}$ is almost identical to that of D_{EE} (Eisenstat trick)

- OpenMP parallelization of the block solver.
  - SSOR in LQCD [S.Fischer, A.Frommer, U.Glässner, Th.Lippert, G.Ritzenhöfer, K.Schilling, CPC 98 (1996), Th.Lippert, P.Comp. 25 (1999)]

- Natural (lexicographical) ordering
- Forward and backward substitutions
- for $(1_{EE}+\omega L_{EE})^{-1}, (1_{EE}+\omega U_{EE})^{-1}$
- have recursive data dependency.
- Ex. Local solution on (1) site is used to
- construct the solution on the sites (7)
- and (2).



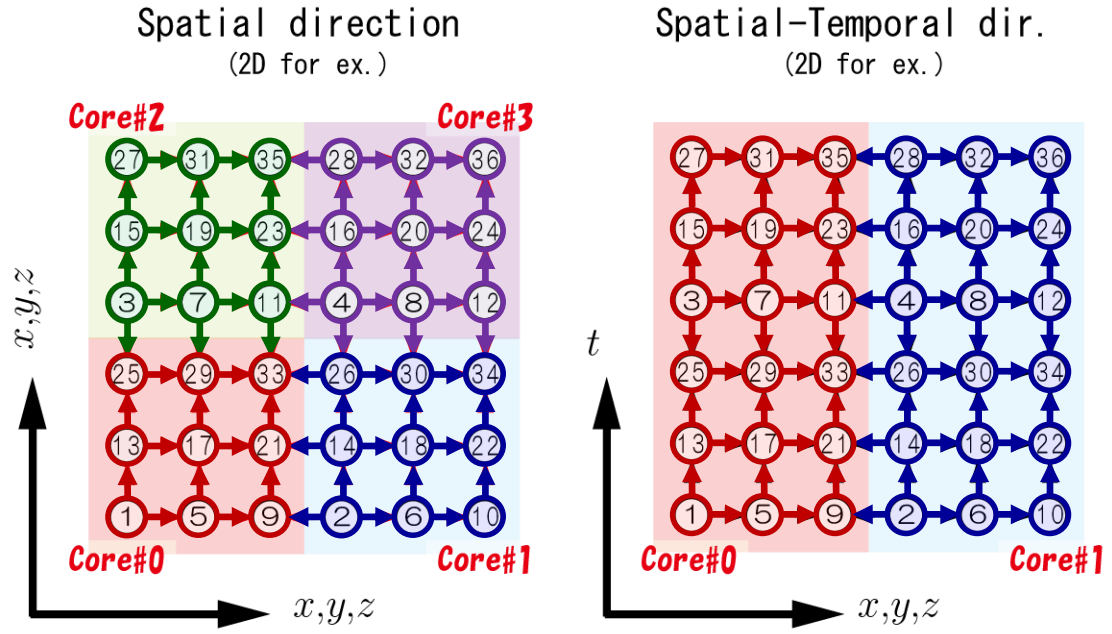
Natural (lexicographical) site ordering in a block

- The data dependency makes the OpenMP threading difficult.
  - Hyper plane ordering. : Task imbalance, list vector data access… will spoil the high potentiality of the K cpu.
- Blocked natural ordering. : Our choice. [Any small block is superior to the even/odd site ordering. Cf. Fischer et. al.]

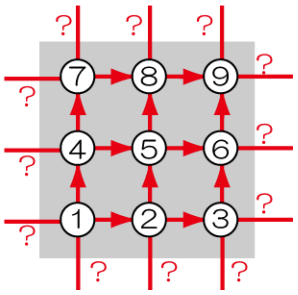
4. Choice of the Block solver in a domain

- SSOR ordering for 8 core OpenMP threading.
  - We divide the block into 16 sub-blocks.
  - Spatial volume is split into 2x2x2 sub-bloks.
    - Each block is assigned to 1 core.
  - Temporal direction is divided to 2.
    - Unrolling the two blocks in a core.

- Each core has the natural ordering
- Each core can solve forward/backward substitution *almost* independently.
- The surface sites of the sub-blocks have core dependency and load imbalance.



Sub-Blocked Natural (local-lexicographical) site ordering in a block.
(arrows indicate data dependency)



Ordering in a sub-block(core).
The data dependency of the surface sites depend on the location of the sub-block.

We implement this ordering for the block solver.

4. Choice of the Block solver in a domain

- We optimize the following Single precision Kernel:

$(DM_{SAP})z = b$: SAP preconditioned BiCGStab(single precision)

(DM_{SAP}): SAP preconditioner

$A_{EE}(\approx D_{EE}^{-1})$: sub-blocked SSOR iteration for the Block inversion.

D_{EE}: Domain restricted operation

D_{EO} (sender): Domain connected operation (sender side)

D_{EO} (receiver): Domain connected operation (receiver side)

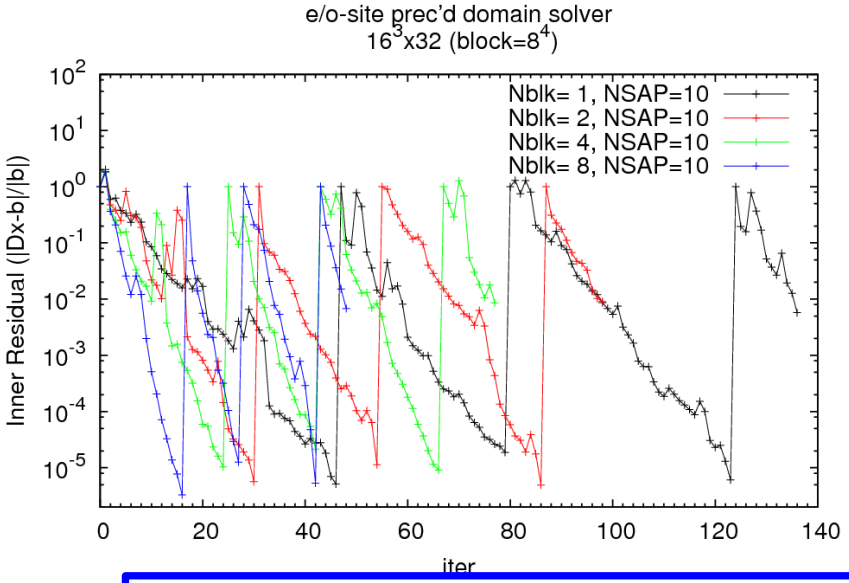
- We optimize them using the SIMD instructions and unrolling to fully utilize the 256 FP registers of the SPARC64 VIIIfx CPU.
- We hide the communication of D_{EO}, D_{OE} behind the computation of D_{EE}, D_{OO}
- We skip the details of these optimization.

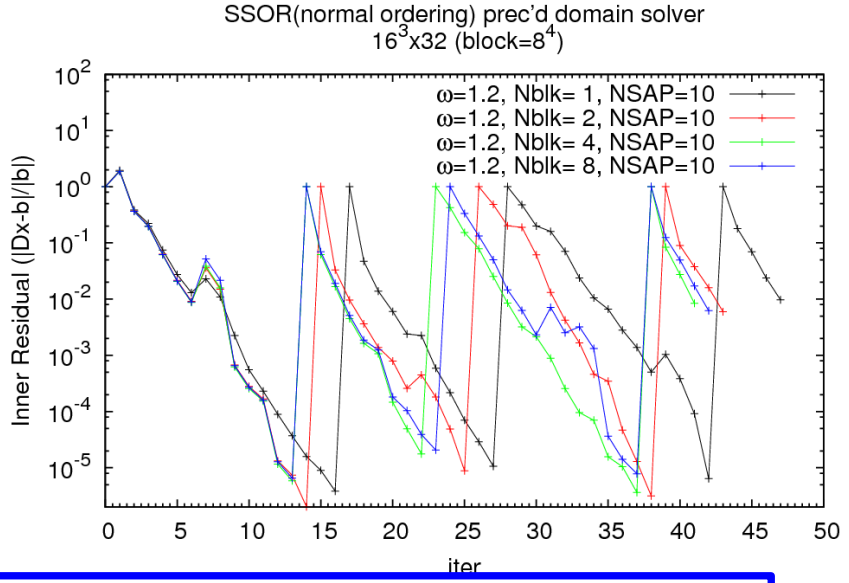
5. Results

- Effect of the domain solver in the SAP prec.
  - We compare the three orderings for the domain solver on T2K Tsukuba (Intel CPU) before implementing it for the K computer:

(1) Even/odd site ordering, (2)SSOR site ordering, (3)Blocked site ordering (2-sub blocks).

– Solver Tests on a quenched 16^3x32 lattice



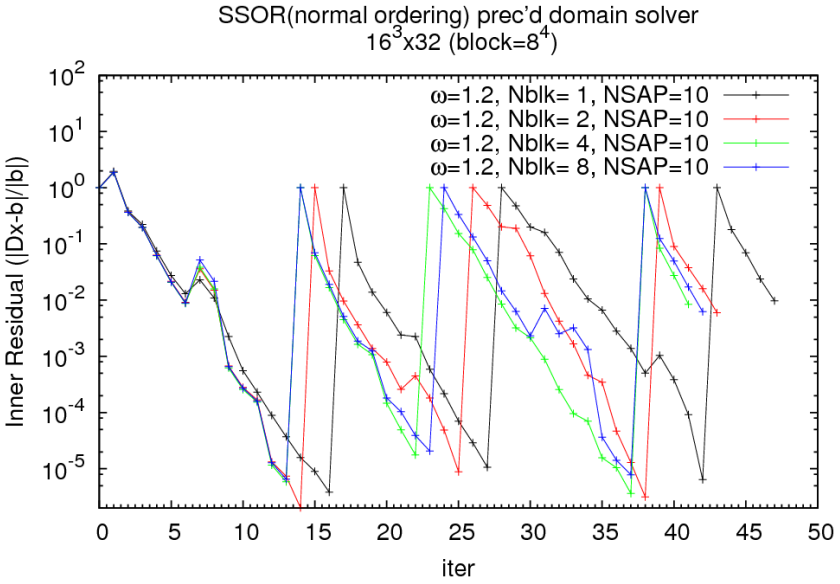


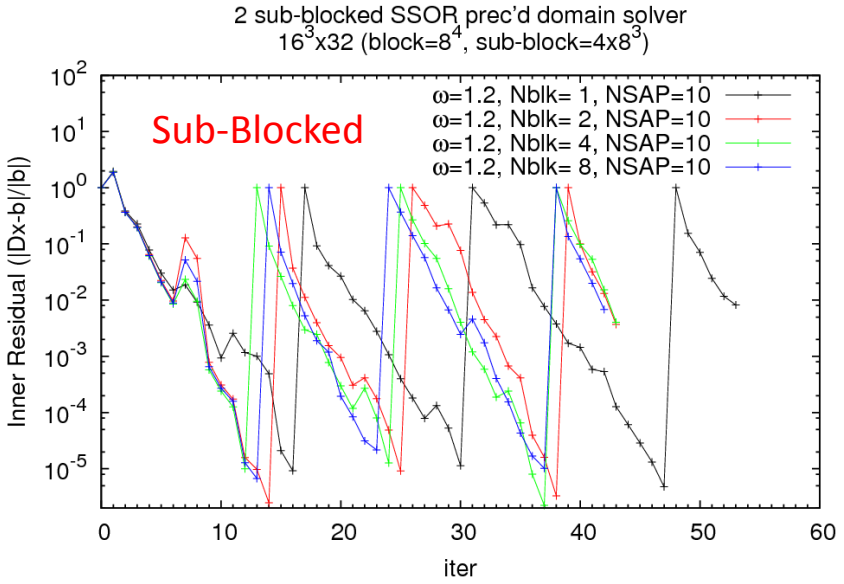
SAP with Even/odd site preconditioned block solver has a poor performance compared to that with the SSOR natural ordering.(Known results)
How about the sub-blocked SSOR preconditioner?

- Effect of the domain solver in the SAP prec.
  - We compare the three orderings for the domain solver on T2K Tsukuba (Intel CPU) before implementing it for the K computer:

(1) Even/odd site ordering, (2)SSOR site ordering, (3)Blocked site ordering (2-sub blocks).

– Solver Tests on a quenched 16^3x32 lattice



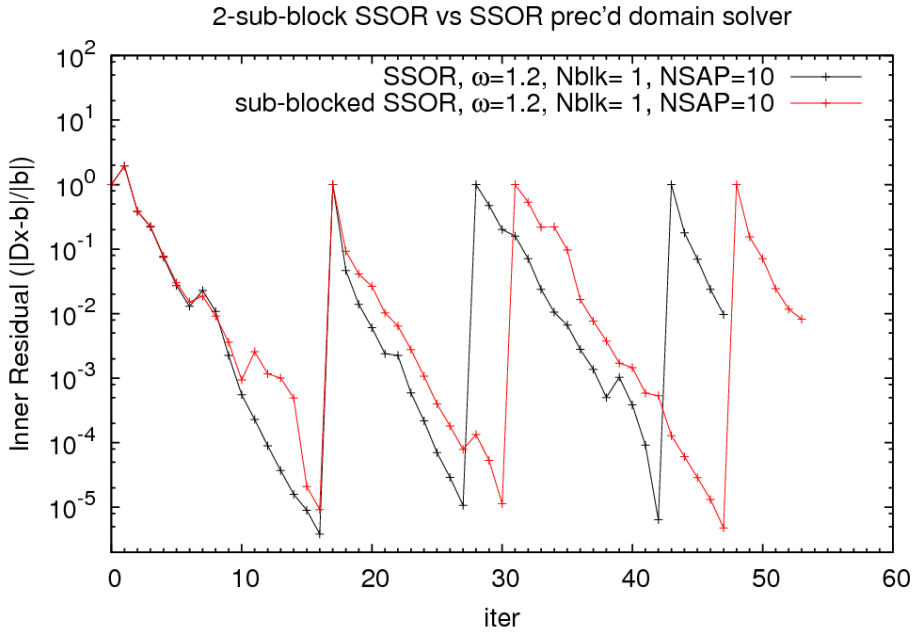


Two-sub blocking increases the iteration counts for the convergence.
The degradation is seems to be little and we can parallelize it with OpenMP.

- Effect of the domain solver in the SAP prec.
  - We compare the three orderings for the domain solver on T2K Tsukuba (Intel CPU) before implementing it for the K computer:

(1) Even/odd site ordering, (2)SSOR site ordering, (3)Blocked site ordering (2-sub blocks).

– Solver Tests on a quenched 16^3x32 lattice



Two-sub blocking increases the iteration counts for the convergence.
The degradation is seems to be little and we can parallelize it with OpenMP.
We decided to implement sub-blocked SSOR for the K computer.

5. Results

- Benchmark tests on the K computer
  - We have measured the performance of the (single precision) Kernel routines:

$(DM_{SAP})z = b$: SAP preconditioned BiCGStab (single precision)

(DM_{SAP}): SAP preconditioner

$A_{EE}(\approx D_{EE}^{-1})$: sub-blocked SSOR iteration for the Block inversion.

D_{EE}: Domain restricted operation

D_{EO} (sender): Domain connected operation (sender side)

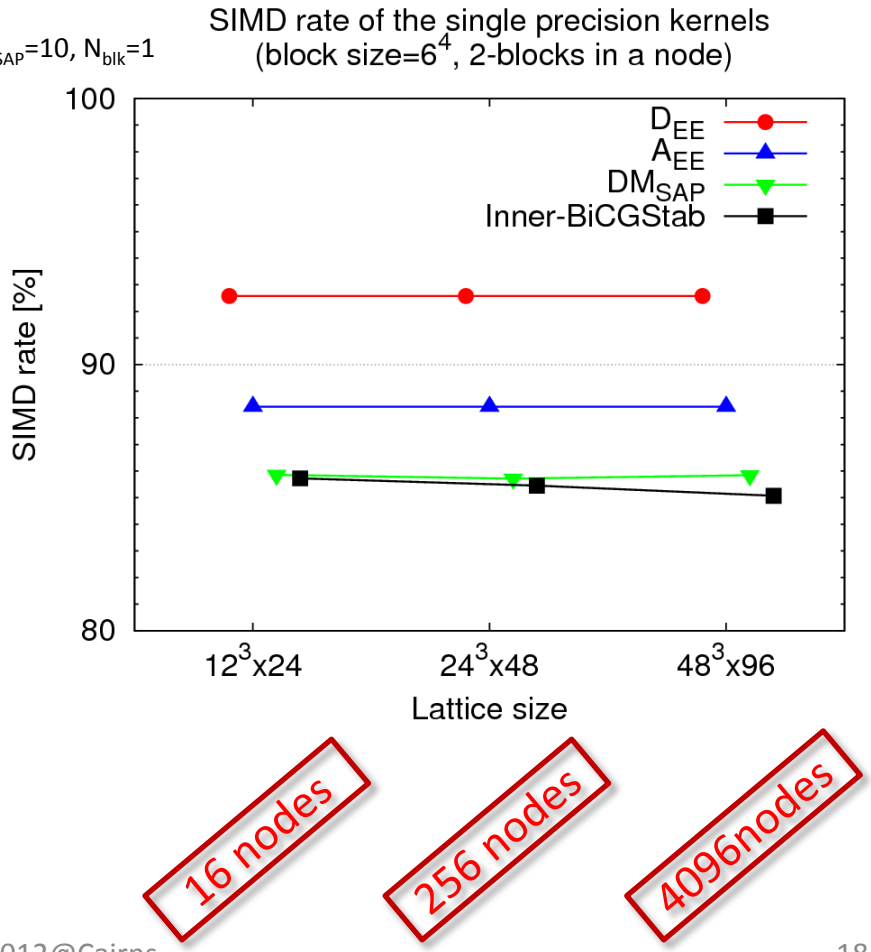
D_{EO} (receiver): Domain connected operation (receiver side)

  - 6^4 block size, 6^3x12 sites in a node. (fixed)
  - Weak scaling test for 12^3x24(16nodes), 24^3x48(256 nodes), and 48^3x96 (4096 nodes) lattices. Solver iteration is fixed.
  - The performance is measured with the profiler and the number contains
    - redundant fp op's from SU(3) reconstruction and Spin projection with FMA (z,x-dirs).
    - This increases the fp op's by about 20% for hopping kernels.

5. Results

- Benchmark tests on the K computer
  - SIMD rate in the executed instructions

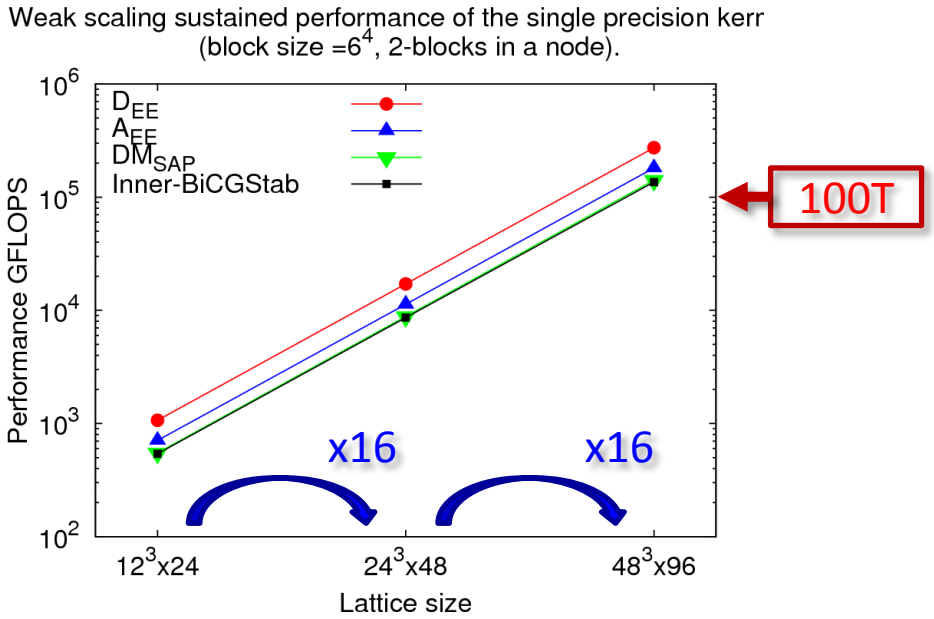
- Over 80% instructions are SIMDized and executed.
- Block restricted D_{EE} has high SIMD rate.
- DM_{SAP} and Inner-BiCGStab are well SIMDized.



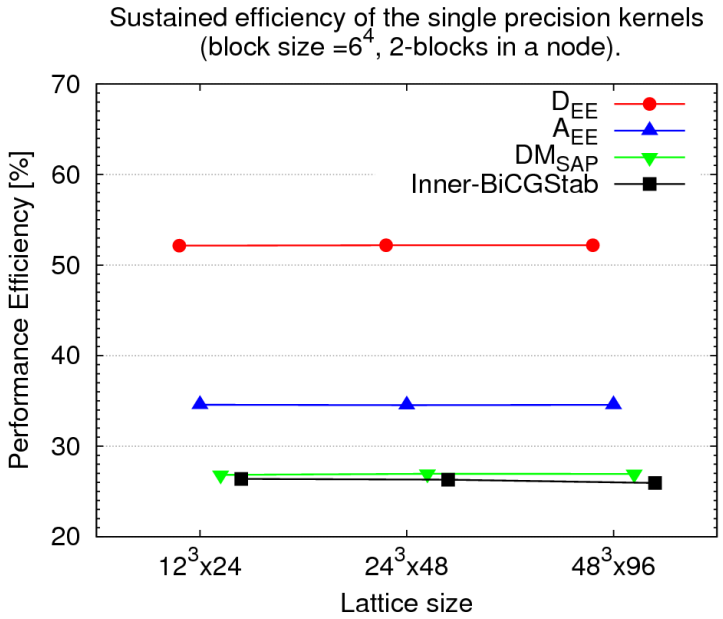
5. Results

- Benchmark tests on the K computer

– Performance and efficiency Weak scaling



100T



- Performance weak scaling is good. Efficiency is kept at ~26% for the inner solver.
- Domain restricted Kernel D_{EE} has over 50 % efficiency. This number contains redundant fp op's from SU(3) reconstruction and spin projection with FMA. Effective flops is multiplied by x 0.8.
- Efficiency reduction for A_{EE} comes from the load imbalance of the forward/backward solvers in the sub-blocked SSOR. Tradeoff between efficiency and dependency (parallelism).

6. Summary

- The blocked SSOR preconditioner still has a good property as a preconditioner than the even/odd-site preconditioner.
- We have implemented the blocked SSOR for the SAP domain solver for the K computer.
- With the blocked SSOR we could utilize the 8-cores of the K computer with the OpenMP threading.
- After optimizing the single precision kernels of the quark solver using basic techniques (SIMDization, loop unrolling), we benchmarked the solver kernels on the K computer.
- Results
  - The domain restricted kernel (D_{EE}) has 50% efficiency of the peak performance.
  - The blocked SSOR kernel (A_{EE}) has a less efficiency due to the load imbalance.
  - The total performance of the single precision BiCGStab solver is at ~26% efficiency and scales ideally in the weak scaling test.
- Full code optimization (including Double precision part) almost completes. Benchmarking has not yet been done.

Backups

- Communication hiding using
  - MPI_Isend/MPI_Irecv/MPI_Wait
  - in the SAP preconditioner DM_{SAP}

Communication hiding

- The SAP preconditioner is built up with the *DK* operation.
- We tune the *DK* performance further using the communication hiding technique.
- The domain connecting operation can be done simultaneously with the domain restricted operation.

DK multiplication on a vector *y*

$$v = (DK)y$$

$$\Downarrow$$

$$x_E = D_{EE}^{-1} y_E$$
$$x_O = D_{OO}^{-1}(y_O - D_{OE}x_E)$$
$$v_E = D_{EE}x_E + D_{EO}x_O$$
$$v_O = D_{OO}x_O + D_{OE}x_E$$

Overlap the computation and communication

$$x_E = D_{EE}^{-1} y_E$$
$$w = D_{OE}x_E \quad \text{(send)}$$
$$v_E = D_{EE}x_E$$
$$f = D_{OE}x_E \quad \text{(recv)}$$
$$w = y_O - f$$
$$x_O = D_{OO}^{-1} w$$
$$w = D_{EO}x_O \quad \text{(send)}$$
$$v_O = D_{OO}x_O + f$$
$$v_E = v_E + D_{EO}x_O \quad \text{(recv)}$$