AICS TECHNICAL REPORT

No. 2014-001

Block BiCGStab for lattice QCD on the K computer

By

H. Suno*, Y. Nakamura, K.-I. Ishikawa, and Y. Kuramashi

RIKEN Advanced Institute for Computational Science

* Corresponding Author E-mail: suno@riken.jp

Submitted on 11/08/2014
Accepted on 03/10/2014

Block BiCGStab for lattice QCD on the K computer

H. Suno*

RIKEN Advanced Institute for Computational Science, Kobe 650-0047, Japan and RIKEN Nishina Center, Wako 351-0198, Japan

Y. Nakamura

RIKEN Advanced Institute for Computational Science, Kobe 650-0047, Japan

K.-I. Ishikawa

Graduate School of Science, Hiroshima University, Hiroshima 739-8526, Japan

Y. Kuramashi

Graduate School of Pure and Applied Science,

University of Tsukuba, Ibaraki 305-8571, Japan,

Center for Computational Science, University of Tsukuba, Ibaraki 305-8577, Japan,

RIKEN Advanced Institute for Computational Science, Kobe 650-0047, Japan

(Dated: August 11, 2014)

Abstract

We have developed a block version of the Bi-Conjugate Gradient Stabilized (BiCGStab) solver for treating the Wilson-Dirac equation with multiple right-hand sides in lattice QCD employing $O(a)$-improved Wilson fermions. Code development and optimization are carried out on the K computer at RIKEN Advanced Institute for Computational Science (AICS). We implement the QR decomposition, a preconditioner based on the domain decomposed Schwarz alternating procedure (SAP) and the symmetric successive over-relaxation (SSOR) iteration with *locally lexicographic* ordering for the inversion within the domain. The block BiCGStab solver, written in single precision, is included as a preconditioning step for an outer BiCGStab solver in double precision. Numerical test is performed on the lattice size of 96^4 using 2048 nodes of the K computer. The performance of a computational kernel exceeds 50% efficiency, and the single precision block BiCGStab solver shows more than $\sim 30\%$ sustained efficiency.

*Electronic address: `suno@riken.jp`

I. INTRODUCTION

The physical point simulation has been one of the primary issues in the first principle calculation of lattice QCD in the recent years. A great effort has been paid to reduce the computational cost for the configuration generation with light quark masses. The algorithmic improvements together with rapid increase of the computational power actually allow us to carry out a direct full QCD simulation on the physical up and down quark masses [1, 2]. While previous improvements mostly concerns the configuration generation, growing effort has been also paid to algorithmic improvements for the measurement of physical observables, so that we are now able to obtain the results for physical quantities at the physical up, down and strange quark masses [3]. Yet it still remains necessary to refine the results reducing the systematic errors and to challenge computationally demanding problems, for example, calculation of the disconnected diagrams.

The main difficulty in lattice QCD simulations resides in solving the Wilson-Dirac equation, which is computationally expensive near the physical up and down quark masses and must be repeated many times both in the configuration generation and measurement of physical observables on given configurations. This can be done in an efficient way only by using a Krylov subspace method, such as the Bi-Conjugate Gradient Stabilized (BiCGStab) method. The performance of the BiCGStab is usually improved with a suitable preconditioning technique such as the Schwarz alternating procedure (SAP) preconditioner proposed by Lüscher, which is applied to the domain-decomposed lattice [4]. The performance is further improved by the nested BiCGStab algorithm with an inner-outer strategy, in which another Krylov subspace method is implemented as a preconditioner[5, 6].

In addition, the Wilson-Dirac equation must be solved with multiple sources in the measurement of physical observables on given configurations: twelve in the simplest case and $O(10-100)$ for the stochastic technique. These are typical examples for differential equations with multiple right-hand sides. For this type of equations, it is well known that the block Krylov subspace solvers works successfully in reducing the computational cost [7, 8]. Since the Wilson-Dirac matrix in lattice QCD is non-Hermitian, we might expect the block BiCGStab algorithm [9] to be applicable in a straightforward manner. Recently, new algorithms, the block BiCGGR and block BiCGStab were found to show significant improvements for this problem [10, 11].

In this work we first develop and optimize an inner block BiCGStab solver in single precision for the K computer at the RIKEN Advanced Institute for Computational Science. Systematic performance tests are carried out for different computational kernels. The solver is used as a preconditioning step for an outer BiCGStab solver in double precision, where different right-hand sides are treated independently. Numerical tests are performed with a 96^4 lattice using 2048 nodes of the K computer.

This report is organized as follows. We explain the block BiCGStab algorithm in detail in Sec. II. Code development and tuning are explained in Sec. III. The results of the numerical test are presented in Sec. IV. Conclusions and discussions are summarized in Sec. V.

II. ALGORITHM

The lattice QCD is defined on a hypercubic four-dimensional lattice with the three-dimensional spatial extent $L_x \times L_y \times L_z$ and the temporal extent L_t. The fields are defined on the four-dimensional lattice sites n with periodic boundary conditions. We define two types of fields on the lattice. One is the gauge field represented by $(U_\mu(n))^{a,b}$ with four-dimensional direction indices $\mu = 1, 2, 3, 4$ and color indices $a, b = 1, 2, 3$, thus which is a 3×3 matrix assigned on each link. The other is the quark field $(q(n))^a_\alpha$ that locates on each site carrying the Dirac indices $\alpha = 1, 2, 3, 4$. The $O(a)$-improved Wilson-Dirac operator preconditioned with the clover term is represented by the matrix elements

$$\begin{aligned} D^{a,b}_{\alpha,\beta}(n,m) \;=\; & \delta^{a,b}\delta_{\alpha,\beta}\delta(n,m) - \kappa F^{a,c}_{\alpha,\gamma}(n)\sum_{\mu=1}^{4}[(1-\gamma_\mu)_{\gamma,\beta}(U_\mu(n))^{c,b}\delta(n+\hat{\mu},m) \\ & + (1+\gamma_\mu)_{\gamma,\beta}((U_\mu(m))^{b,c})^*\delta(n-\hat{\mu},m)], \end{aligned} \tag{1}$$

where $\hat{\mu}$ denotes the unit vector in the μ direction in the four-dimensional space-time, $F(n)$ the inverse clover term $(1 - (c_{\rm SW}\kappa/2)\sigma_{\mu\nu}F_{\mu\nu}(n))^{-1}$ with $c_{\rm SW}$ being an adjusting parameter for the $O(a)$-improvement. Our gamma matrices γ_μ are defined as

$$\gamma_1 = \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & -i & 0 \\ 0 & i & 0 & 0 \\ i & 0 & 0 & 0 \end{pmatrix}, \gamma_2 = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix},$$

$$\gamma_3 = \begin{pmatrix} 0 & 0 & -i & 0 \\ 0 & 0 & 0 & i \\ i & 0 & 0 & 0 \\ 0 & -i & 0 & 0 \end{pmatrix}, \gamma_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{2}$$

The $O(a)$-improved Wilson-Dirac operator defined above is a complex non-Hermitian square matrix, where only 51 out of $L_x \times L_y \times L_z \times L_t \times 3 \times 4$ entries in each row have nonzero values. The matrix is fairly sparse in current numerical simulations with $L_{x,y,z,t} \sim O(10 - 100)$.

Calculations of the physical quantities requires the solution of linear equations with multiple right-hand sides:

$$\sum_{m,\beta,b} D^{a,b}_{\alpha,\beta}(n,m) x^{(i)}(m)^b_\beta = b^{(i)}(n)^a_\alpha, \tag{3}$$

where $b^{(i)}$ $(i = 1, \cdots L)$ represents the i-th source vector. This is expressed in a simple form:

$$DX = B, \tag{4}$$

where D is an $N \times N$ complex sparse non-Hermitian matrix, X and B are $N \times L$ complex rectangular matrices given by

$$X = [x^{(1)}, \ldots, x^{(i)}, \ldots, x^{(L)}], \tag{5}$$

$$B = [b^{(1)}, \ldots, b^{(i)}, \ldots, b^{(L)}]. \tag{6}$$

In the Wilson-Dirac equation, the matrix dimension is given by $N = L_x \times L_y \times L_z \times L_t \times 3 \times 4$, whereas we designate as L the number of the right-hand sides corresponding to the different source vectors in lattice QCD.

Pseudocode for the block BiCGStab algorithm is described in Algorithm 1 below, which is improved with two ingredients. One is the QR decomposition with a modified Gram-Schmidt orthogonalization method, which improves numerical accuracy since each span works effectively to search approximative solutions. The other is Lüscher's SAP preconditioner [4], which is applied to the domain-decomposed lattice: The Wilson-Dirac matrix D is decomposed in the 2×2 blocked matrix form

$$D = \begin{pmatrix} D_{EE} & D_{EO} \\ D_{OE} & D_{OO} \end{pmatrix}, \tag{7}$$

where the subscripts E and O denote the even and odd domains, respectively. The SAP preconditioner M_{SAP} is computed as

$$M_{\mathrm{SAP}} = K \sum_{j=0}^{N_{\mathrm{SAP}}-1} (1 - DK)^j \tag{8}$$

with

$$K = \begin{pmatrix} A_{EE} & 0 \\ -A_{OO}D_{OE}A_{EE} & A_{OO} \end{pmatrix}, \tag{9}$$

where A_{EE} (A_{OO}) can be any approximation for $(D_{EE})^{-1}$ ($(D_{OO})^{-1}$). When A_{EE} and A_{OO} are exact, DK becomes block triangular and is expected to be well conditioned. In the case $|DK| < 1$, M_{SAP} converges to D^{-1} when $N_{\mathrm{SAP}} \to \infty$. The approximate inverse of the operator A_{EE} (A_{OO}) is the most important part in the SAP. The exactness is not required for the SAP, however, the better approximation with less computational cost is preferred for A_{EE} (A_{OO}). We adopt the symmetric successive over-relaxation (SSOR) method [12], which is derived by decomposing the original operator into the sum of an upper and a lower triangle matrices. We have then:

$$A_{EE} = (1 - \omega U_{EE})^{-1} \left[\sum_{j=0}^{N_{\mathrm{SSOR}}} (1 - A_{EE}^{\mathrm{SSOR}})^j \right] (1 - \omega L_{EE})^{-1} \tag{10}$$

with

$$A_{EE}^{\mathrm{SSOR}} = \frac{1}{\omega}[(1 - \omega L_{EE})^{-1} + (1 - \omega U_{EE})^{-1} + (\omega - 2)(1 - \omega L_{EE})^{-1}(1 - \omega U_{EE})^{-1}], \tag{11}$$

where L_{EE} and U_{EE} are the lower and upper triangle matrices corresponding to the forward and backward hopping terms, respectively. To extract 8 core parallelism for the K computer (for the computer architecture, see Sec. III), we further divide the block into 16 sub-blocks via the *locally-lexicographical* ordering (*ll*-ordering) as described in detail in Refs. [12, 13]. Since $(1 - \omega L_{EE})$ and $(1 - \omega U_{EE})$ are triangular matrices, their inverse can be directly calculated via forward and backward substitutions, respectively. Since more than 80% of the CPU time is spent for the forward/backward substitutions, optimization must concentrate upon these parts.

III. CODE DEVELOPMENT AND OPTIMIZATION FOR THE K COMPUTER

Our code development and optimization are carried out for the K computer at the RIKEN Advanced Institute for Computational Science. The machine consists of 82944 computa-

Algorithm 1 Block BiCGStab algorithm $(D, M_{\text{SAP}}, B, \epsilon_i)$.

```
 1: initial guess $X \in \mathbb{C}^{N\times L}$,
 2: compute $R = B - DX$,
 3: set $P = R$,
 4: choose $\tilde{R} \in \mathbb{C}^{N\times L}$,
 5: while $(|r^{(i)}|/|b^{(i)}|) > \epsilon_i$ do
 6:     QR decomposition $P = Q\gamma$, $P \leftarrow Q$,
 7:     $U = M_{\text{SAP}}P$,
 8:     $V = DU$,
 9:     solve $(\tilde{R}^H V)\alpha = \tilde{R}^H R$ for $\alpha$,
10:     $R \leftarrow R - V\alpha$,
11:     $X \leftarrow X + U\alpha$,
12:     $S = M_{\text{SAP}}R$,
13:     $Z = DS$,
14:     $\zeta = \text{Tr}(Z_i^H R_i)/\text{Tr}(Z_i^H Z_i)$,
15:     $X \leftarrow X + \zeta S$,
16:     $R \leftarrow R - \zeta Z$,
17:     solve $(\tilde{R}^H V)\beta = -\tilde{R}^H Z$ for $\beta$,
18:     $P \leftarrow R + (P - \zeta V)\beta$,
19: end while
```

tional nodes and 5184 I/O nodes connected by the so-called "Tofu" network, providing 11.28 Pflops of computing capability. The Tofu network topology is six-dimensional one with 3D-mesh times 3D-torus shape. Each node has a single 2.0GHz SPARC64 VIIIfx processor equipping 8 cores with SIMD enabled 256 registers, 6MB shared L2 cache and 16GB of memory. The L1 cache sizes per each core are 32KB/2WAY (instruction) and 32KB/2WAY (data).

We have developed a block version of the forward and backward solvers constructing A_{EE} (or A_{OO}) in the SSOR for the multiple right-hand sides, based on the non-block version[13]. The forward and backward kernels constitute the most important part in the SSOR. Both consists of several four-fold nested loops: three spatial direction x, y, z-loops and one temporal t-loop. They are typically written as

```
for $i_x = 0$ to $N_x - 1$ do
  for $i_y = 0$ to $N_y - 1$ do
    for $i_z = 0$ to $N_z - 1$ do
      OpenMP Barrier
      for $i_t = 0$ to $N_t - 1$ do
        if $i_t > 0$ then
          $y''(n)^a_\alpha = (1+\gamma_4)_{\alpha,\beta}((U(n-\hat{t}))^{b,a})^* y(n-\hat{t})^b_\beta$
        end if
        $\ldots$
        $y''(n)^a_\alpha \Leftarrow F(n)^{a,b}_{\alpha,\beta} y''(n)^b_\beta$
        $y'(n)^a_\alpha = y(n)^a_\alpha + \kappa y''(n)^a_\alpha$
      end for
    end for
  end for
end for,
```

or as the same way with all the loops in the reversed order, where $N_{x,y,z,t}$ denote the number of lattice sites in x, y, z, t directions inside the SAP block. Note that an explicit OpenMP barrier is inserted because of the recurrence dependency among the OpenMP threads. There are several conditional branches for site location like $i_t > 0$, $i_x > 0$ and $i_x < N_x - 1$, etc... inside the t-loop. The question arises as to where to put the loop running over different right-hand side vectors in Eq. (4). Our choice is just inside the t-loop like (we designate as case (a)):

```
for $i_t = 0$ to $N_t - 1$ do
  for $i = 0$ to $L - 1$ do
    if $i_t > 0$ then
      $y''(n)^a_\alpha = (1+\gamma_4)_{\alpha,\beta}((U(n-\hat{t}))^{b,a})^* y^{(i)}(n-\hat{t})^b_\beta$
    end if
    $\ldots$
    $y''(n)^a_\alpha \Leftarrow F(n)^{a,b}_{\alpha,\beta} y''(n)^b_\beta$
    $y'^{(i)}(n)^a_\alpha = y(n)^a_\alpha + \kappa y''(n)^a_\alpha$
  end for
end for,
```

or even inside the `if` statements (we designate as case (b)):

```
for $i_t = 0$ to $N_t - 1$ do
  if $i_t > 0$ then
    for $i = 0$ to $L - 1$ do
      $y''^{(i)}(n)^a_\alpha = (1 + \gamma_4)_{\alpha,\beta}((U(n - \hat{t}))^{b,a})^* y^{(i)}(n - \hat{t})^b_\beta$
    end for
  end if
  ...
  for $i = 0$ to $L - 1$ do
    $y''^{(i)}(n)^a_\alpha \Leftarrow F(n)^{a,b}_{\alpha,\beta} y''^{(i)}(n)^b_\beta$
    $y'^{(i)}(n)^a_\alpha = y^{(i)}(n)^a_\alpha + \kappa y''^{(i)}(n)^a_\alpha$
  end for
end for.
```

We have carried out performance tests for these two cases with lattice sizes $12^3 \times 24$ and $24^3 \times 48$, where the SAP block sizes correspond to 6^4 and 12^4, respectively, on 16 nodes of the K computer choosing $L = 1, 4$ and 12 for the number of the right-hand sides. For all the performance tests presented in this Section, we choose the hopping and improvement parameters $(\kappa, c_{\mathrm{SW}}) = (0.123, 1.0)$. Table I shows efficiency, SIMD rate and several time consuming parts. Although it is difficult to observe the general trend because of the complexity of their behavior with respect to L, the efficiency and SIMD rate increase well with L. We notice that (OpenMP) barrier synchronization wait and integer cache wait decrease with L. This is because the L-loop does not include any OpenMP barrier statement inside it. For all the values of L the better performance is found for case (a) than for case (b); We obtain a better SIMD rate and less time for floating-point cache wait. We should also note that case (a) has less loop overhead than case (b) so that SIMDization and software pipelining should work better. We have hence chosen to put the L-loop inside the t-loop and outside the `if` statements, as in case (a). Since the D_{EE} and D_{OO} kernels involve a similar four-fold nested loop with several `if` statements inside, though without OpenMP barrier synchronization, we have inserted the L-loop inside the t-loop and outside `if` statements.

We have measured the performance of the A_{EE} and D_{EE} kernels as well as the DM_{SAP}

TABLE I: Results for the performance tests of the block A_{EE} (A_{OO}) kernel on 16 nodes of the K computer. We have tested two cases: The L-loop is located just inside the t-loop in case (a), while inside the `if` statements in case (b). LA stands for load access.

Case	Lattice	L	Effic.	SIMD	4 instr.	2/3 instr.	1 instr.	Barrier sync.	Fl. pt. op.	Fl. pt. cache	Int. cache	Fl.pt. memory
	size		[%]	rate[%]	commit[%]	commit[%]	commit[%]	wait[%]	LA wait[%]	LA wait[%]	LA wait[%]	LA wait[%]
(a)	$12^3 \times 24$	1	31	78.1	11.4	23.2	15.2	10.5	11.6	14.3	5.4	0.0
		4	39	86.8	14.1	25.1	22.2	10.3	15.5	6.8	2.3	0.2
		12	40	88.0	13.8	27.6	20.4	9.5	12.1	11.4	1.1	1.1
	$24^3 \times 48$	1	29	77.0	10.6	21.7	15.6	9.3	10.1	13.3	4.6	9.5
		4	34	84.9	11.7	23.5	17.7	9.5	11.5	16.1	2.0	2.6
		12	36	87.8	13.0	23.9	19.0	5.7	13.3	18.1	0.9	1.6
(b)	$12^3 \times 24$	1	19	59.4	8.5	20.3	18.0	10.8	10.0	14.6	5.8	0.0
		4	25	73.0	7.7	23.9	21.8	10.4	9.3	14.7	2.7	0.2
		12	26	74.7	7.8	23.4	22.6	8.8	8.9	16.8	1.1	0.9
	$24^3 \times 48$	1	17	60.9	8.1	17.2	16.1	7.6	8.8	14.8	3.6	10.5
		4	21	73.6	7.4	19.8	18.0	7.7	8.3	22.5	1.8	6.4
		12	24	75.0	8.8	19.9	20.5	5.3	8.3	24.2	0.7	4.7

multiplication and the block BiCGStab algorithm with two lattice sizes of $12^3 \times 24$ and $24^3 \times 48$ on 16 nodes of the K computer. Figure 1 presents the performance efficiencies with respect to the floating point number operations for these kernels as a function of L. All the performance efficiencies are increased with L. We should note that the difference between the block BiCGStab and DM_{SAP} slightly becomes larger as L increases. This is mostly due to the QR decomposition, which requires more **MPI_Allreduce** calls for increased number of L. We have also carried out weak-scaling performance tests by increasing the number of nodes with the SAP block size fixed to 6^4 and 12^4 at $L = 12$: We have chosen lattices sizes of $24^3 \times 48$ on 256 nodes, 48^4 on 2048 nodes, $48^3 \times 96$ on 256 nodes, and 96^4 on 2048 nodes. The results are shown in Fig. 2. We observe that the performance drops little with increasing number of nodes for most of the computational kernels. Even in the case of the block BiCGStab with the SAP block size of 6^4, which shows the worst scaling behavior, the efficiency is still better than 30% for the largest lattice size. The numerical results of these performance tests are summarized in Table II, together with the measured SIMD rate and several time consuming parts.

The nested BiCGStab algorithm is designed to be flexible against the preconditioner changing iteration by iteration [5, 6], where the outer BiCGStab contains an inner BiCGStab as the flexible preconditioner. The flexibility ensures the double precision accuracy of the solution vector even if we used the single precision for the inner BiCGStab solver. In our work, the block BiCGStab solver described above is used as a preconditioner or inner solver for the outer solver, the former in single precision and the latter in double precision. Pseudocode for the outer solver is described in Algorithm 2. Note that the preconditioning steps at lines 8 and 14 in Algorithm 2 consist of solving approximatively the Dirac equation with multiple right-hand sides $DV = P$ and $DV = R$ by using the single precision block BiCGStab, which is described in Algorithm 1. We can quickly obtain an approximative solution to the Wilson-Dirac equation by using the block BiCGStab algorithm.

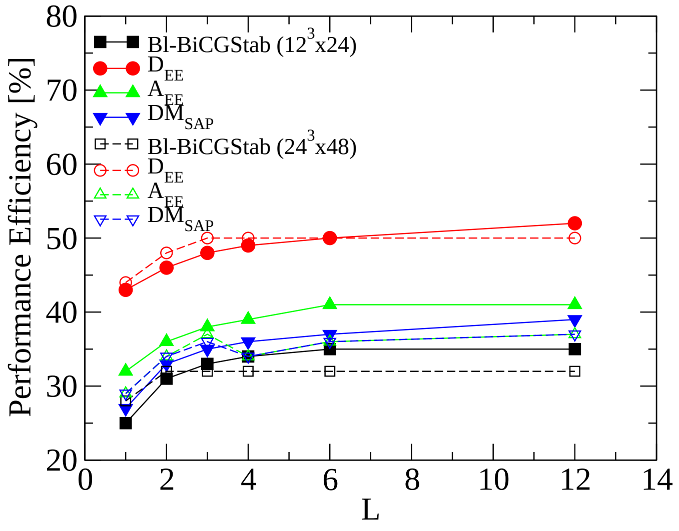

FIG. 1: Performance efficiencies of D_{EE}, A_{EE}, DM_{SAP} kernels and the block BiCGStab algorithm as a function of L with $12^3 \times 24$ (closed) and $24^3 \times 48$ (open) lattices on 16 nodes of the K computer.

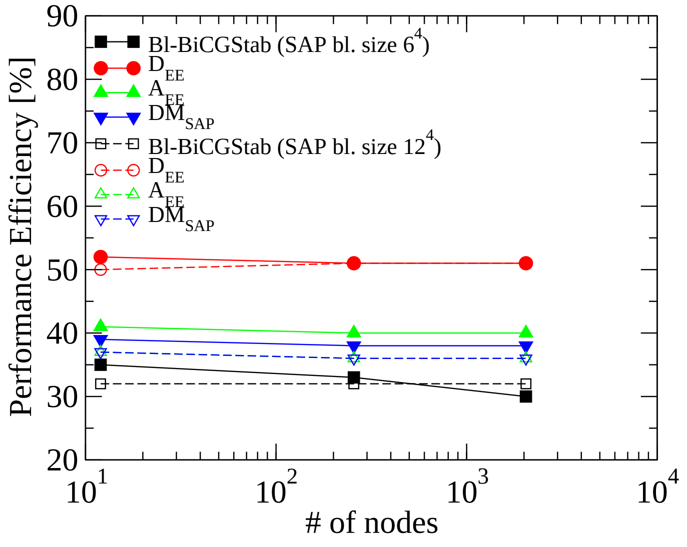

FIG. 2: Performance efficiencies of D_{EE}, A_{EE}, DM_{SAP} kernels and the block BiCGStab algorithm as a function of the number of nodes for two different SAP block size 6^4 (closed) and 12^4 (open). The number of right-hand sides is fixed to $L = 12$.

TABLE II: Results for the performance tests of the block BiCGStab kernel with different lattice sizes on different numbers of nodes of the K computer. The SAP block sizes are 6^4 in case (a) and 12^4 in case (b). LA stands for load access. These tests have been carried out with the number $L = 12$ of right-hand sides, but for 96^4 lattice we also include the results for $L = 1$ and $L = 4$ for the sake of comparison.

Case	Lattice	number	Effic.	SIMD	4 instr.	2/3 instr.	1 instr.	Barrier sync.	Fl. pt. op.	Fl. pt. cache	Int. cache	Fl.pt. memory
	size	of nodes	[%]	rate[%]	commit[%]	commit[%]	commit[%]	wait[%]	LA wait[%]	LA wait[%]	LA wait[%]	LA wait[%]
(a)	$12^3 \times 24$	16	35	84.6	13.9	24.4	19.2	7.6	10.5	13.0	2.7	1.9
	$24^3 \times 48$	256	33	80.7	9.7	24.5	20.1	7.2	9.7	11.9	4.3	1.8
	48^4	2048	30	83.6	17.1	22.1	18.1	6.2	8.9	11.2	7.2	1.7
(b)	$24^3 \times 48$	16	33	85.1	13.0	21.5	17.7	4.7	11.2	17.9	1.0	7.2
	$48^3 \times 96$	256	32	85.1	13.0	21.3	17.7	4.6	11.1	18.1	1.3	7.2
	96^4	2048	32	84.9	12.0	21.1	17.9	4.7	11.0	17.8	1.8	7.0
	$96^{4\,a}$	2048	27	76.0	12.4	19.5	15.5	8.3	10.3	13.2	5.3	8.6
	$96^{4\,b}$	2048	31	85.0	15.6	21.0	16.6	6.2	10.7	17.6	3.0	4.2

[a]Calculated for $L = 1$
[b]Calculated for $L = 4$

Algorithm 2 Outer BiCGStab algorithm

```
 1: for $i = 0$ to $L - 1$ do
 2:     initial guess $x^{(i)} \in \mathbb{C}^N$,
 3:     compute $r^{(i)} = b^{(i)} - Dx^{(i)}$,
 4:     set $p^{(i)} = r^{(i)}$,
 5:     choose $\tilde{r}^{(i)}$ such that $(\tilde{r}^{(i)}, r^{(i)}) \neq 0$,
 6: end for
 7: while $\max_i(|r^{(i)}|/|b^{(i)}|) > \epsilon^{\text{outer}}$ do
 8:     solve $D[v^{(1)}, \ldots, v^{(L)}] = [p^{(1)}, \ldots, p^{(L)}]$ by the block BiCGStab
 9:     for $i = 0$ to $L - 1$ do
10:         $\alpha^{(i)} = (\tilde{r}^{(i)}, r^{(i)})/(\tilde{r}^{(i)}, Dv^{(i)})$,
11:         $x^{(i)} \leftarrow x^{(i)} + \alpha^{(i)} v^{(i)}$,
12:         $r^{(i)} \leftarrow r^{(i)} - \alpha^{(i)} Dv^{(i)}$,
13:     end for
14:     solve $D[v^{(1)}, \ldots, v^{(L)}] = [r^{(1)}, \ldots, r^{(L)}]$ by the block BiCGStab
15:     for $i = 0$ to $L - 1$ do
16:         $\zeta^{(i)} = (Dv^{(i)}, r^{(i)})/(Dv^{(i)}, Dv^{(i)})$,
17:         $x^{(i)} \leftarrow x^{(i)} + \zeta^{(i)} v^{(i)}$,
18:         $r^{(i)} \leftarrow r^{(i)} - \zeta^{(i)} Dv^{(i)}$,
19:         $\beta^{(i)} = (\alpha^{(i)}/\zeta^{(i)}) \cdot (\tilde{r}^{(i)}, r^{(i)})/(\tilde{r}^{(i)}, r'^{(i)})$,
20:         $p^{(i)} \leftarrow r^{(i)} + \beta_k^{(i)}(p^{(i)} - \zeta^{(i)} Dv^{(i)})$,
21:         $r'^{(i)} = r^{(i)}$
22:     end for
23: end while
```

IV. NUMERICAL TEST

Our numerical tests are performed using 2048 nodes of the K computer. We use a thermalized configuration generated at almost the physical point with $(\kappa_{ud}, \kappa_s) = (0.126117, 0.124790)$ on 96^4 lattice in $2 + 1$ flavor lattice QCD employing the nonperturbatively $O(a)$-improved Wilson quark action with $c_{\rm SW} = 1.11$ and the Iwasaki gauge action at $\beta = 1.82$. Stout smearing procedure is performed with the smearing parameters $N_{\rm stout} = 6$ and $\alpha = 0.1$. We choose the hopping parameter $\kappa = 0.126125$ for the Wilson-Dirac equation, and $N_{\rm SAP} = 5$ with 12^4 SAP block size following Ref. [1]. Parameters for the SSOR method are fixed with $N_{\rm SSOR} = 1$ and $\omega = 1.24$. The stopping criterion is set to be $\max_i(|r^{(i)}|/|b^{(i)}|) \leq 10^{-14}$ for the outer BiCGStab, while the stopping criterion $(|r^{(i)}|/|b^{(i)}|) \leq \epsilon_i^{\rm inner}(i = 1, 2, ..., L)$ for the inner block BiCGStab is set by the outer BiCGStab from iteration to iteration ranging from $\epsilon_i^{\rm inner} \approx 10^{-7}$ to 10^{-3}. We have used so-called "noise sources" generated with random numbers for multiple right-hand sides.

Figure 3 presents a representative case for the residual norm as a function of the total number of inner matrix-vector multiplications (IMVMs) for the block BiCGstab with $L = 1, 2, 3, 4, 6$ and 12 in addition to the residual norm of the outer BiCGStab solver. The residual norm of the inner block BiCGStab decreases well along with the iteration, and once the stopping condition is satisfied, the algorithm is switched from the inner BiCGStab to the outer one and returns again to the inner one in order to resume the next preconditioning step with the residual norm initialized. Although for some cases the inner residual norm decreases more quickly as L increases, the total number of iterations for convergence does not diminish so drastically as observed in Refs. [6, 11], where the block BiCGStab is employed as the outer solver. A possible reason is a rather loose stopping criterion compared to the one used in Refs. [6, 11], where the required number of iterations seems to decrease much more conspicuously as a function of L with a more stringent stopping criterion. Table III shows the L dependence of the computational cost to solve the Wilson-Dirac equation as well as the maximum and minimum values of the true residuals. We present the averaged values over the $(12/L)$ samples of L noise source vectors for the number of inner matrix-vector multiplications and the execution time. The computational cost divided by L is found to decrease as L increases. We do not have a problem of the convergence failure. In Table III we also show the performance and efficiency of the block BiCGStab solver. It turns out that we obtain more than 30% efficiency with $L \geq 2$ and the performance goes beyond 85TFLOPS for several values of L. This implies that we have a clear gain in using the block BiCGStab instead of its non-blocked version, where the maximum efficiency is about 26%.

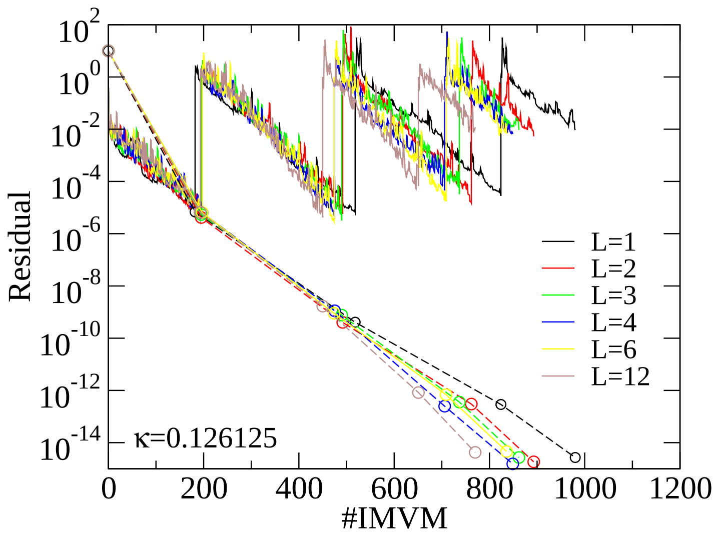

FIG. 3: Respresentative behavior of the residual norm as a function of the number of inner matrix-vector multiplications (IMVMs), together with the residual norm of the outer solver. The number of right-hand side vectors are chosen to be $L = 1, 2, 3, 4, 6$, and 12.

TABLE III: L dependence of the number of inner matrix-vector multiplications (IMVMs) and the execution time to solve the Wilson-Dirac equation. True residual is evaluated after the relative residual in the outer solver reaches the tolerance of 10^{-14}.

L	#IMVM	Time [s]	Time/L [s]	True residual		Performance	Efficiency
				Max	Min	[TFLOPS]	[%]
1	915.3	65.2	65.2	4.99×10^{-15}	7.24×10^{-16}	71	27
2	913.0	128.4	64.2	2.69×10^{-15}	4.59×10^{-16}	81	31
3	876.8	193.8	64.6	4.49×10^{-15}	4.20×10^{-16}	86	33
4	869.7	235.6	58.9	2.21×10^{-15}	4.09×10^{-16}	82	31
6	856.5	349.3	58.2	4.62×10^{-15}	4.48×10^{-16}	85	32
12	770.0	669.0	55.8	4.25×10^{-15}	6.52×10^{-16}	83	32

V. SUMMARY

In this work we have developed the single precision block BiCGStab solver for treating the $O(a)$-improved Wilson-Dirac equation with multiple right-hand sides in lattice QCD. The code development is carried out for the K computer and the optimization for the computation kernels shows the improvement in performance depending on the SAP block size, the number of compute nodes, and the number of right-hand sides L. The performance of the block BiCGStab solver is found to be more than 30% efficiency. With the block BiCGStab solver used as the preconditioning step for the outer BiCGStab solver in double precision, we have succeeded in test calculations with 96^4 lattice size on 2048 nodes of the K computer. The computational cost reduces with increasing number of L. It may be possible to make a further improvement for the QR decomposition in future.

Acknowledgments

All the results are obtained by using the K computer at the RIKEN Advanced Institute for Computational Science.

[1] S. Aoki, K.-I. Ishikawa, N. Ishizuka, T. Izubuchi, D. Kadoh, K. Kanaya, Y. Kuramashi, Y. Namekawa, M. Okawa, Y. Taniguchi, et al. (PACS-CS Collaboration), Phys. Rev. D **79**, 034503 (2009).

[2] Y. Kuramashi, PoS **Lattice 2008**, 018 (2008).

[3] S. Aoki, K.-I. Ishikawa, N. Ishizuka, T. Izubuchi, D. Kadoh, K. Kanaya, Y. Kuramashi, Y. Namekawa, M. Okawa, Y. Taniguchi, et al. (PACS-CS Collaboration), Phys. Rev. D **81**, 074503 (2010).

[4] M. Lüscher, JHEP **0305**, 052 (2003).

[5] J. A. Vogel, App. Math. Comput. **188**, 226 (2007).

[6] H. Tadano and T. Sakurai, LSSC'07, Lec. Notes Comput. Sci. **4818**, 721 (2008).

[7] D. O'Leary, Lenear Algebra Appl. **29**, 293 (1980).

[8] A. Nikishin and A. Yu. Yeremin, SIAM J. Matrix Anal. Appl. **16**, 1135 (1995).

[9] A. E. Guennouni, K. Jbilou, and H. Sadok, Electron. Trans. Numer. Anal. **16**, 129 (2003).

[10] T. Sakurai, H. Tadano, and Y. Kuramashi, Comput. Phys. Commun. **181**, 113 (2010).

[11] Y. Nakamura, K.-I. Ishikawa, Y. Kuramashi, T. Sakurai, and H. Tadano, Comput. Phys. Commun. **183**, 34 (2012).

[12] S. Fischer, A. Frommer, U. Glässner, T. Lippert, G. Ritzenhöfer, and K. Schilling, Comput.

Phys. Commun. **98**, 20 (1996).

[13] T. Boku, K.-I. Ishikawa, Y. Kuramashi, K. Minami, Y. Nakamura, F. Shoji, D. Takahashi, M. Terai, A. Ukawa, and T. Yoshié, PoS **Lattice 2012**, 188 (2012).