Electronic Preprint for Journal of Information Processing Vol.24 No.1

Regular Paper

Hybrid Numerical Solvers for Massively Parallel Eigenvalue Computations and Their Benchmark with Electronic Structure Calculations

Hiroto Imachi[1,a)] Takeo Hoshi[1]

Received: April 21, 2015, Accepted: May 26, 2015

Abstract: Optimally hybrid numerical solvers were constructed for massively parallel generalized eigenvalue problems (GEP). The strong scaling benchmark was carried out on the K computer and other supercomputers for electronic structure calculation problems in the matrix sizes of $M = 10^4 - 10^6$ with up to 10^5 cores. The procedure of GEP is decomposed into the two subprocedures of the reducer to the standard eigenvalue problem (SEP) and the solver of SEP. A hybrid solver is constructed, when a routine is chosen for each subprocedure from the three parallel solver libraries of ScaLAPACK, ELPA and EigenExa. The hybrid solvers with the two newer libraries, ELPA and EigenExa, give better benchmark results than the conventional ScaLAPACK library. The detailed analysis on the results implies that the reducer can be a bottleneck in next-generation (exa-scale) supercomputers, which provides guidance for future research. The code was developed as a middleware and a mini-application and will appear online.

Keywords: massively parallel numerical library, generalized eigenvalue problem, electronic structure calculation, ELPA, EigenExa, the K computer, mini-application

1. Introduction

Numerical linear algebraic solvers for large matrices have strong needs among various applications with current and next-generation supercomputers. Nowadays ScaLAPACK [1], [2] *1 is the *de facto* standard solver library for parallel computations, but several routines give severe bottlenecks in the computational speed with current massively parallel architectures. Novel solver libraries were proposed so as to overcome the bottlenecks. Since the performance of numerical routines varies significantly with problems and architectures, the best performance is achieved, when one constructs an optimal 'hybrid' among the libraries.

The concept of a hybrid solver is illustrated in **Fig. 1**. It is a numerical middleware and has a unique data interface to real applications. One can choose the optimal workflow for each problem without any programming effort.

The present paper focuses on dense-matrix solvers for generalized eigenvalue problems (GEPs) in the form of

$$A\boldsymbol{y}_k = \lambda_k B\boldsymbol{y}_k \tag{1}$$

with the given $M \times M$ real-symmetric matrices of A and B. The matrix B is positive definite. The eigenvalues $\{\lambda_k\}$ and the eigenvectors $\{\boldsymbol{y}_k\}$ will be calculated. The computational cost is $O(M^3)$ or is proportional to M^3. The present hybrid solvers are constructed among ScaLAPACK and the two newer libraries of ELPA [3], [4], [5] *2, and EigenExa [6], [7], [8], [9]. The ELPA and EigenExa libraries are written in Fortran and appeared in the 2000s for efficient massively parallel computations.

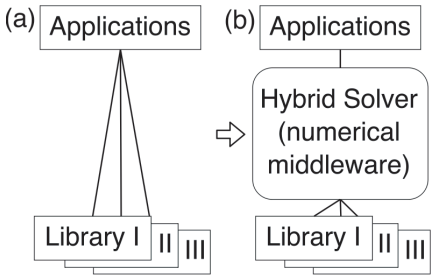

Fig. 1 Concept of a hybrid solver; Structure of the program code (a) without and (b) with hybrid solver or numerical middleware.

The present paper is organized as follows: Section 2 explains the background of the electronic structure calculation. Section 3 describes the mathematical foundation. Sections 4 and 5 are devoted to the benchmark results and discussions, respectively. The summary and future outlook appear in Section 6.

2. Background

2.1 Large-scale Electronic Structure Calculations

The GEP of Eq. (1) gives the mathematical foundation of electronic structure calculations or quantum mechanical calculations of materials, in which an electron is treated as a quantum mechanical 'wave'. The input matrix A or B of Eq. (1) is called a Hamiltonian or an overlap matrix, respectively. An eigenvalue of $\{\lambda_k\}$ is the energy of one electron and an eigenvector of $\{\boldsymbol{y}_k\}$ specifies the wavefunction or the shape of an electronic 'wave'.

[1] Tottori University, JST-CREST, Tottori 680–8552, Japan
[a)] D14T1001B@edu.tottori-u.ac.jp

*1 ScaLAPACK = Scalable Linear Algebra PACKage
*2 ELPA = Eigenvalue soLvers for Petascale Applications

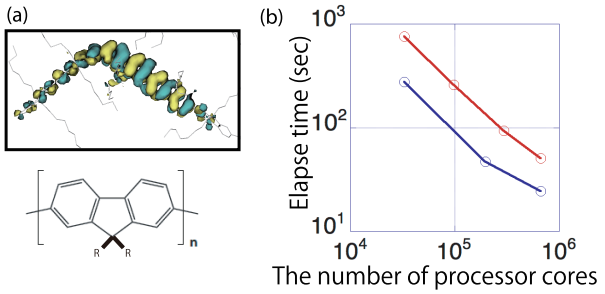

Fig. 2 (a) The upper panel is a π-type electronic wavefunction in an amorphous-like conjugated polymer (poly-((9,9) dioctyl fluorine)). The lower panel shows the atomic structure (R≡C_8H_{17}) [10]. (b) Strong scaling plot by ELSES for one-hundred-million-atoms calculations on the K computer [10], [11]. The calculated materials are a nano-composite carbon solid (the upper line) and the amorphous-like conjugated polymer (the lower line). The number of used processor nodes are from P = 4,096 to 82,944 (full nodes of the K computer).

Figure 2 (a) shows an example of the wavefunction. A typical number of the required eigenvalues is, at least, on the order of the number of the electrons or the atoms in calculated materials. See the ELPA paper [3] for a review, because ELPA was developed under tight collaboration with the electronic structure calculation society.

Here, our motivation is explained. The present authors developed a large-scale quantum material simulator called ELSES [*3] [12], [13]. The theories are explained in Refs. [13], [14] and the reference therein. The matrices are based on the real-space atomic-orbital representation, and the matrix size M is nearly proportional to the number of atoms N ($M \propto N$). The simulations mainly use novel 'order-N' linear-algebraic methods in which the computational cost is 'order-N' ($O(N)$) or is proportional to the number of atoms N. Their mathematical foundation is sparse-matrix (Krylov-subspace) solvers. An efficient massively parallel computation is found in Fig. 2, a strong scaling benchmark on the K computer [10], [11] with one hundred million atoms or one-hundred-nanometer scale materials. The simulated materials are a nano-composite carbon solid with N = 103,219,200 or M = 412,876,800 [10] and an amorphous-like conjugated polymer with N = 102,238,848 or M = 230,776,128 [11].

The present dense-matrix solvers are complement to the order-N solvers, because the order-n solvers give approximate solutions, while the dense-matrix solvers give numerically exact ones with a heavier ($O(M^3)$) computational cost. The use of the two methods will lead us to fruitful research. The exact solutions are important, for example, when the system has many nearly degenerated eigen pairs and one would like to distinguish them. The exact solutions are important also as reference data for the development of fine approximate solvers.

The matrices of A and B in the present benchmark appear on 'ELSES Matrix Library' [15]. The Library is the collection of the matrix data generated by ELSES for material simulations. The benchmark was carried out with the data files of 'NCCS430080', 'VCNT22500' 'VCNT90000' and 'VCNT1008000' for the matrix sizes of M = 22,500, M = 90,000, M = 430,080, M = 1,008,000, respectively. Files with the size of 0.5 GB or larger are uploaded as a set of split files for user convenience.

The physical origin of the matrices is explained briefly. The files in the present benchmark are carbon materials within modeled tight-binding-form theories based on *ab initio* calculations. The matrix of 'NCCS430080' appears in our material research on a nano-composite carbon solid (NCCS) [16]. An sp-orbital form [17] is used and the system contains $N = M/4$ = 107,520 atoms. The other files are generated for thermally vibrated single-wall carbon nanotubes (VCNTs) within a supercell. An spd-orbital form [18] is used and each system contains $N = M/9$ atoms. The VCNT systems were prepared, so as to generate matrices systematically in different sizes with similar eigenvalue distributions. We used these matrices for the investigation on π-electron materials with the present dense-matrix solver and the order-N solver [*4].

3. The Hybrid Solvers

A hybrid solver is constructed, when a routine is chosen for each subprocedure from ScaLAPACK, EigenExa and ELPA. The code was developed as a general middleware that can be connected not only to ELSES but also to any real application software, as in Fig. 1. A mini-application was also developed and used in the present benchmark. In the benchmark, ScaLAPACK was used as a built-in library on each machine. EigenExa version 2.2a [*5] and ELPA version 2014.06.001 were used. ELPA and EigenExa call some ScaLAPACK routines.

3.1 Mathematical Formulation

The GEP of Eq. (1) can be written in a matrix form of

$$AY = BY\Lambda, \tag{2}$$

where the matrix $\Lambda \equiv \mathrm{diag}(\lambda_1, \lambda_2, \dots)$ is diagonal and the matrix $Y \equiv (\boldsymbol{y}_1\, \boldsymbol{y}_2\, \cdots)$ satisfies $Y^T BY = I$. In the solvers, the GEP of Eq. (1) is reduced to a standard eigenvalue problem (SEP) of

$$A'Z = Z\Lambda, \tag{3}$$

where the reduced matrix A' is real symmetric [20] and the matrix of $Z \equiv (\boldsymbol{z}_1\, \boldsymbol{z}_2\, \cdots)$ contains eigenvectors of A'. The reduction procedure can be achieved, when the Cholesky factorization of B gives the Cholesky factor U as an upper triangle matrix:

$$B = U^T U. \tag{4}$$

The reduced matrix A' is defined by

$$A' = U^{-T} A U^{-1}. \tag{5}$$

The eigenvectors of the GEP, written as $Y \equiv (\boldsymbol{y}_1\, \boldsymbol{y}_2\, \cdots)$, are calculated from those of the SEP by

$$Y = U^{-1} Z. \tag{6}$$

This procedure is usually called backward transformation.

[*3] ELSES = Extra-Large-Scale Electronic Structure calculation.

[*4] The present matrices are sparse, which does not lose the generality of the benchmark, since the cost of the dense matrix solver is not dependent on the number of non-zero elements of the matrix.

[*5] The present EigenExa package does not include the GEP solver. The GEP solver routine for EigenExa in the present paper is that of KMATH_EIGEN_GEV version 2.2b [19] that shares the SEP solver routine with the EigenExa package.

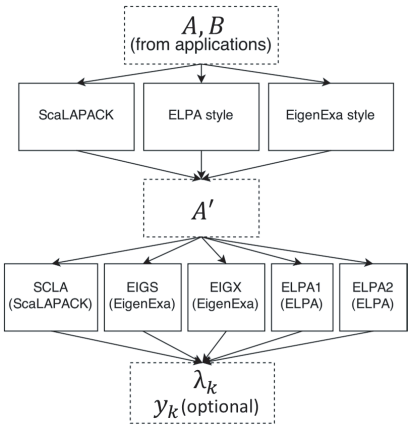

Fig. 3 Workflow of the hybrid GEP solver.

The GEP solver is decomposed into the two subprocedures of (a) the solver of the SEP in Eq. (3) and (b) the reduction from the GEP to the SEP ($(A, B) \Rightarrow A'$) and the backward transformation ($Z \Rightarrow Y$). The subprocedures (a) and (b) are called 'SEP solver' and 'reducer', respectively, and require $O(M^3)$ operations.

Figure 3 summarizes the workflows of the possible hybrid solvers. A hybrid solver is constructed, when one chooses the routines for (a) the SEP solver and (b) the reducer, respectively.

For (a) the SEP solver, five routines are found in the base libraries; one routine is a ScaLAPACK routine (routine name in the code: 'pdsyevd') that uses the conventional tridiagonalization algorithm [21]. The ELPA or EigenExa library contains a SEP solver routine based on the tridiagonalization algorithm. The routine in ELPA is called 'ELPA1' (routine name in the code: 'solve_evp_real') in this paper, as in the original paper [3], and the one in EigenExa called 'Eigen_s' or 'EIGS' (routine name in the code: 'eigen_s'). ELPA and EigenExa also contain the novel SEP solvers based on the narrow-band reduction algorithms without the conventional tridiagonalization procedure. The solvers are called 'ELPA2' (routine name in the code: 'solve_evp_real_2stage') for the ELPA routine and 'Eigen_sx' or 'EIGX' (routine name in the code: 'eigen_sx') for the EigenExa routine in this paper. See the papers [4], [8] for details.

For (b) the reducer, three routines are found in the base libraries and are called ScaLAPACK style, ELPA style, and EigenExa style reducers in this paper. In the ScaLAPACK style, the Cholesky factorization, Eq. (4) is carried out and then the reduced matrix A', defined in Eq. (5), is generated by a recursive algorithm (routine name 'pdsygst') without explicit calculation of U^{-1} nor U^{-T}. Details of the recursive algorithm are explained, for example in Ref. [22]. In the ELPA style, the Cholesky factorization (routine name: 'cholesky_real') is carried out, as in the ScaLAPACK style, and the reduced matrix A' is generated by the explicit calculation of the inverse (triangular) matrix $R \equiv U^{-1}$ (routine names: 'invert_trm_real') and the explicit successive matrix multiplication of $A' = (R^T A)R$ (routine names: 'mult_at_b_real') [3] [*6]. In the EigenExa style, the Cholesky factorization is not used. Instead, the SEP for the matrix B

$$BW = WD, \tag{7}$$

is solved by the SEP solver (Eigen_sx), with the diagonal matrix of $D \equiv \mathrm{diag}(d_1, d_2, ...)$ and the unitary matrix of $W \equiv (\boldsymbol{w}_1\, \boldsymbol{w}_2\, ...)$. A reduced SEP in the form of Eq. (3) is obtained by

$$A' = (D^{-1/2}W^T)A(WD^{-1/2}) \tag{8}$$

$$Y = WD^{-1/2}Z, \tag{9}$$

because of $Z = D^{1/2}W^T Y$ and $W^{-T} = W$. Equation (9) is solved by the SEP solver (Eigen_sx).

Though the SEP solver of Eq. (3) requires a larger operation cost than the Cholesky factorization (See Fig. 1 of Ref. [23], for example), the elapse time can not be estimated only from the operation costs among modern supercomputers.

The benchmark of the hybrid GEP solvers was carried out for the eight workflows listed in **Table 1**. In general, a potential issue is the possible overhead of the data conversion process between libraries. This issue will be discussed in Section 5.2.

Table 1 List of the workflows in the benchmark. The routine names for the SEP solver and the reducer are shown for each workflow. Abbreviations are shown within parentheses.

Workflow	SEP solver	Reducer
A	ScaLAPACK (SCLA)	ScaLAPACK (SCLA)
B	Eigen_sx (EIGX)	ScaLAPACK (SCLA)
C	ScaLAPACK (SCLA)	ELPA
D	ELPA2	ELPA
E	ELPA1	ELPA
F	Eigen_s (EIGS)	ELPA
G	Eigen_sx (EIGX)	ELPA
H	Eigen_sx (EIGX)	Eigen_sx (EIGX)

4. Benchmark Result

Strong scaling benchmarks are investigated for the hybrid solvers. The elapse times were measured for (i) the full eigenpair calculation (T_{full}) and (ii) the 'eigenvalue-only' calculation (T_{evo}). In the latter case, the elapse time is ignored for the calculation of the eigenvectors. The two types of calculations are important among electronic structure calculations [3]. The present benchmark ignores small elapse times of the initial procedure for distributed data and the comments on them will appear in Section 5.1.

The benchmark was carried out on three supercomputers; the K computer at Riken, Fujitsu FX10 and SGI Altix ICE 8400EX. The K computer has a single SPARC 64 VIIIfx processor (2.0 GHz, 8-core) on node. The FX10 is Oakleaf-FX of the University of Tokyo. Fujitsu FX10 is the successor of the K computer and has a single SPARC64 IXfx processor (1.848 GHz, 16-core) on each node [*7]. We also used the SGI Altix ICE 8400EX of the Institute for Solid State Physics of the University of Tokyo. It is a cluster of Intel Xeon X5570 (2.93 GHz, 8-core). The byte-per-flop value (B/F) is B/F=0.5, 0.36 or 0.68, for the K computer, FX10 or SGI Altix, respectively. The numbers of used processor nodes P are set to be square numbers ($P = q^2$) except in Section 4.3, since the ELPA paper [3] reported that the choice of a

[*6] The benchmark was carried out in an ELPA style reduction algorithm. The ScaLAPACK routine of 'pdtrmm' is used for the multiplication of the triangular matrix R from right, while a sample code in the ELPA package uses the ELPA routine ('mult_at_b_real'). We ignore the difference, since the elapse time of the above procedure is not dominant.

[*7] Additional options of the K computer and FX10 are explained; we did not specify an MPI process shape on the Tofu interconnect. We used the rank directory feature to alleviate I/O contention.

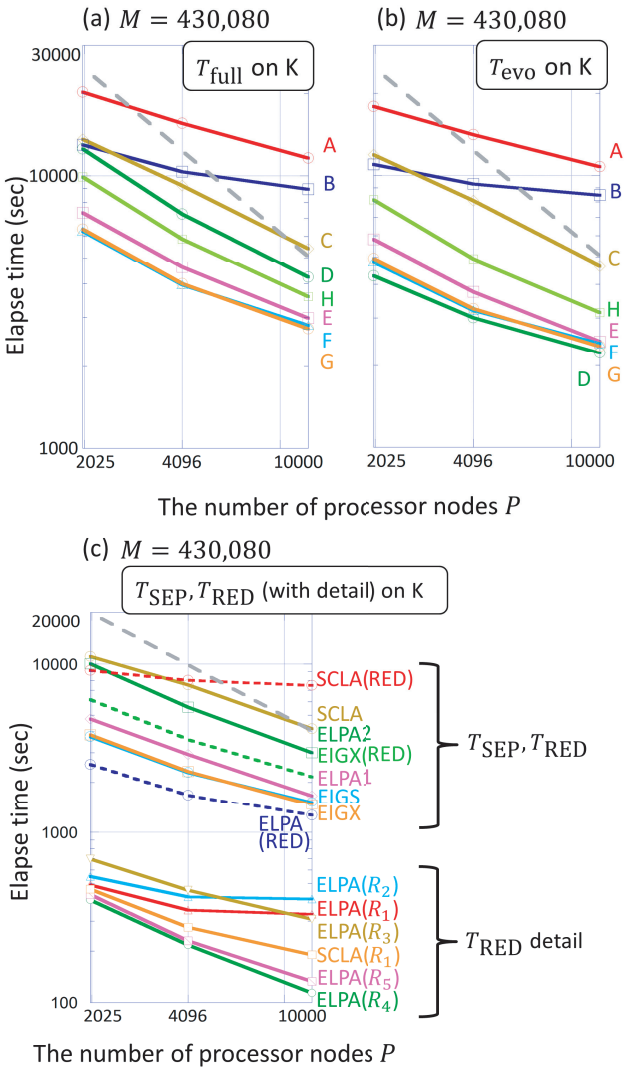

Fig. 4 Results with M = 430,800 on the K computer. The elapse times are plotted with the workflows for the (a) full (T_{full}) and (b) eigenvalue-only (T_{evo}) calculations. (c) The decomposed times for the SEP solver (T_{SEP}) and for the reducer (T_{RED}) are plotted. The routines for the reducers are labeled by '(RED)'. Detailed decomposed times for subprocedures of the ELPA style reducer and the Cholesky decomposition in the ScaLAPACK style reducer are also plotted in (c). The ideal speedup in parallelism is drawn as a dashed gray line.

(near-)square number for P can give better performance.

When the non-traditional SEP solver algorithm of ELPA is used on Altix, one can choose an optimized low-level routine using SSE instructions ('REAL_ELPA_KERNEL_SSE') and a generic routine ('REAL_ELPA_KERNEL_GENERIC') [3]. The optimized code can run only on the Intel-based architectures compatible to SSE instructions and was prepared so as to accelerate the backtransformation subroutine. Among the results on Altix, the 'ELPA2' solver and the workflow D on Altix are those with the optimized routine, while the 'ELPA2'' solver and the workflow D' are those with the generic routine.

4.1 Result with the Matrix Size of M = 430,080

The benchmark with the matrix size of M = 430,080 was carried out for up to P = 10,000 nodes on the K computer. The elapse times for P = 10,000 nodes is shown in **Table 2**. The elapse time for all the cases are shown in **Fig. 4** for the (a) full (T_{full}) or (b) eigenvalue-only (T_{evo}) calculations. The decomposed times are also shown in Fig. 4 (c) for the SEP solver (T_{SEP}) and the reducer (T_{RED}) ($T_{\text{full}} = T_{\text{SEP}} + T_{\text{RED}}$).

Table 3 shows the decomposed time of the SEP solvers for

Table 2 Selected results of the benchmark. The elapse time for the full (eigenpair) calculation (T_{full}) and that for the eigenvalue-only calculation (T_{evo}) with the workflows. The recorded time is the best data among ones with different numbers of the used nodes. The number of used nodes (P) for the best data is shown within parentheses. The best data among the workflows are labelled by '[B]'. The saturated data are labelled by '[S]'. The workflow D' on Altix is that without the SSE optimized routine of the 'ELPA2' SEP solver. See the text for details.

Size M/Machine	WF	T_{full} (sec)	T_{evo} (sec)
1,000,080/FX10	G	39,919 (P = 4,800)	35,103 (P = 4,800)
430,080/K	A	11,634 (P = 10,000)	10,755 (P = 10,000)
	B	8,953 (P = 10,000)	8,465 (P = 10,000)
	C	5,415 (P = 10,000)	4,657 (P = 10,000)
	D	4,242 (P = 10,000)	2,227 (P = 10,000)[B]
	E	2,990 (P = 10,000)	2,457 (P = 10,000)
	F	2,809 (P = 10,000)	2,416 (P = 10,000)
	G	2,734 (P = 10,000)[B]	2,355 (P = 10,000)
	H	3,595 (P = 10,000)	3,147 (P = 10,000)
90,000/K	A	590 (P = 4,096)	551 (P = 4,096)
	B	493 (P = 1,024)[S]	449 (P = 1,024)[S]
	C	318 (P = 4,096)	298 (P = 4,096)
	D	259 (P = 4,096)	190 (P = 4,096)[B]
	E	229 (P = 4,096)[B]	194 (P = 4,096)
	F	233 (P = 4,096)	210 (P = 4,096)
	G	258 (P = 4,096)	240 (P = 4,096)
	H	253 (P=4,096)	236 (P=4,096)
90,000/FX10	A	1,248 (P = 1,369)	1,183 (P = 1,369)
	B	691 (P = 1,024)[S]	648 (P = 1,024)[S]
	C	835 (P = 1,369)	779 (P = 1,369)
	D	339 (P = 1,369)	166 (P = 1,024)[B][S]
	E	262 (P = 1,369)	233 (P = 1,024)[S]
	F	250 (P = 1,369)[B]	222 (P = 1,369)
	G	314 (P = 1,024)[S]	283 (P = 1,024)[S]
	H	484 (P=1,369)	456 (P=1,369)
90,000/Altix	A	1,985 (P = 256)	1,675 (P = 256)
	B	1,883 (P = 256)	1,586 (P = 256)
	C	1,538 (P = 256)	1,240 (P = 256)
	D	1,621 (P = 256)	594 (P = 256)
	D'	2,621 (P = 256)	585 (P = 256)[B]
	E	1,558 (P = 256)	1,287 (P = 256)
	F	1,670 (P = 256)	1,392 (P = 256)
	G	1,453 (P = 256)[B]	1,170 (P = 256)
	H	2,612 (P=256)	2,261 (P=256)
22,500/K	A	65.2 (P = 1,024)	59.6 (P = 256)
	B	45.8 (P = 1,024)[S]	43.2 (P = 1,024)[S]
	C	41.7 (P = 2,025)	37.8 (P = 2,025)
	D	28.4 (P = 2,025)	22.6 (P = 1,024)
	E	28.3 (P = 2,025)[B]	22.6 (P = 1,024)[B]
	F	28.8 (P = 1,024)[S]	26.9 (P = 1,024)[S]
	G	29.7 (P = 1,024)[S]	27.8 (P = 1,024)[S]
	H	39.3(P=1024)[S]	37.5(P=1024)[S]
22,500/FX10	A	126.2 (P = 256)	118.1 (P = 256)
	B	71.3 (P = 256)[S]	67.1 (P = 256)[S]
	C	103.5 (P = 256)[S]	96.3 (P = 256)[S]
	D	30.5 (P = 529)[B]	24.4 (P = 529)[B]
	E	34.3 (P = 256)[S]	31.2 (P = 256)[S]
	F	32.1 (P = 529)	29.4 (P = 529)
	G	45.3 (P = 529)	42.5 (P = 529)
	H	74.9(P=529)	72.2 (P=529)
22,500/Altix	A	51.4 (P = 256)	42.1 (P = 256)
	B	70.0 (P = 256)	50.7 (P = 256)
	C	45.6 (P = 256)	35.5 (P = 256)
	D	41.8 (P = 256)	22.3 (P = 256)[B]
	D'	59.6 (P = 256)	21.8 (P = 256)[B]
	E	32.3 (P = 256)[B]	26.7 (P = 256)
	F	48.5 (P = 256)	37.3 (P = 256)
	G	57.2 (P = 256)	39.6 (P = 256)
	H	71.2 (P=256)	64.1 (P=256)

Table 3 Decomposition of the elapse time (sec) of the SEP solvers with M = 430,080 and P = 10,000. See the text for the subroutine names of 'TRD/BAND', 'D&C' and 'BACK'.

SEP solver	TRD/BAND	D&C	BACK	Total ($T_{\rm SEP}$)
SCLA	3,055	465	633	4,152
ELPA2	966	141	1,892	2,999
ELPA1	1,129	138	400	1,667
EIGS	1,058	196	265	1,521
EIGX	828	390	255	1,473

P = 10,000. A SEP solver routine is decomposed into three subroutines of (i) the tridiagonalization or narrow-band reduction ('TRD/BAND'), (ii) the divide and conquer algorithms for the tridiagonal or narrow-band matrices ('D&C') so as to compute the eigenvalues, and (iii) the backtransformation of eigenvectors ('BACK') so as to compute the eigenvectors of the GEP.

One can observe several features on the results; (I) In the full calculation benchmark (Fig. 4 (a)), the best data, the smallest elapse time, appears in the workflow G for P = 10,000. The workflow G is the hybrid solver that uses the 'Eigen_sx' SEP solver in EigenExa and the ELPA style reducer, since these routines are the best among the SEP solvers and the reducers, respectively, as shown in Fig. 4 (c) and Table 3. In Table 2, the speed ($T_{\rm full}^{-1}$) of the workflow G is approximately four times faster than that of the conventional workflow A (11,634 sec)/(2,734 sec) ≈ 4.3). (II) Figure 4 (c) shows that the ELPA style reducer gives significantly smaller elapse times than those of ScaLAPACK and those of EigenExa. The elapse time for P = 10,000 is $T_{\rm RED}$ = 1,261 sec with the ELPA style reducer and is $T_{\rm RED}$ = 2,157 sec with the EigenExa reducer. The elapse time with the EigenExa reducer is governed by that of the SEP solver for Eq. (7) ($T_{\rm SEP}$ = 1,473 sec in Table 3). (III) In the eigenvalue-only calculation (Fig. 4 (b)), the best data, the smallest elapse time, appears in the workflow D for P = 10,000. The workflow D is the solver that uses the 'ELPA2' SEP solver and the ELPA style reducer and the eigenvector calculation consumes a large elapse time of $T_{\rm vec}$; $T_{\rm vec} \equiv T_{\rm full} - T_{\rm evo}$ = (4,242 sec) − (2,227 sec) = (2,015 sec) in Table 2. The time $T_{\rm vec}$ is contributed mainly by the backward transformation subroutine ($T_{\rm BACK}$ = 1,892 sec) in Table 3, because the backward transformation subroutine in ELPA2 uses a characteristic two-step algorithm (See Section 4.3 of Ref. [3]).

4.2 Benchmark with the Matrix Sizes of M = 90,000, 22,500

The benchmark with the smaller matrix sizes of M = 90,000 and 22,500 are also investigated. The maximum number of used processor nodes is $P_{\rm max}$ = 4,096, 1,039 and 256, on the K computer, FX10, and Altix, respectively [*8]. **Figures 5** and **6** show the data with M = 90,000 and with M = 22,500, respectively. The decomposed times are shown in **Fig. 7**. Table 2 shows the best data for each workflow among the different numbers of used nodes. The results will help general simulation researchers to choose the solver and the number of used nodes, since the elapse times in Table 2 are less than a half hour and such calculations are popular 'regular class' jobs among systematic investigations [*9].

Here, the results are discussed; (I) Table 2 shows that the small-

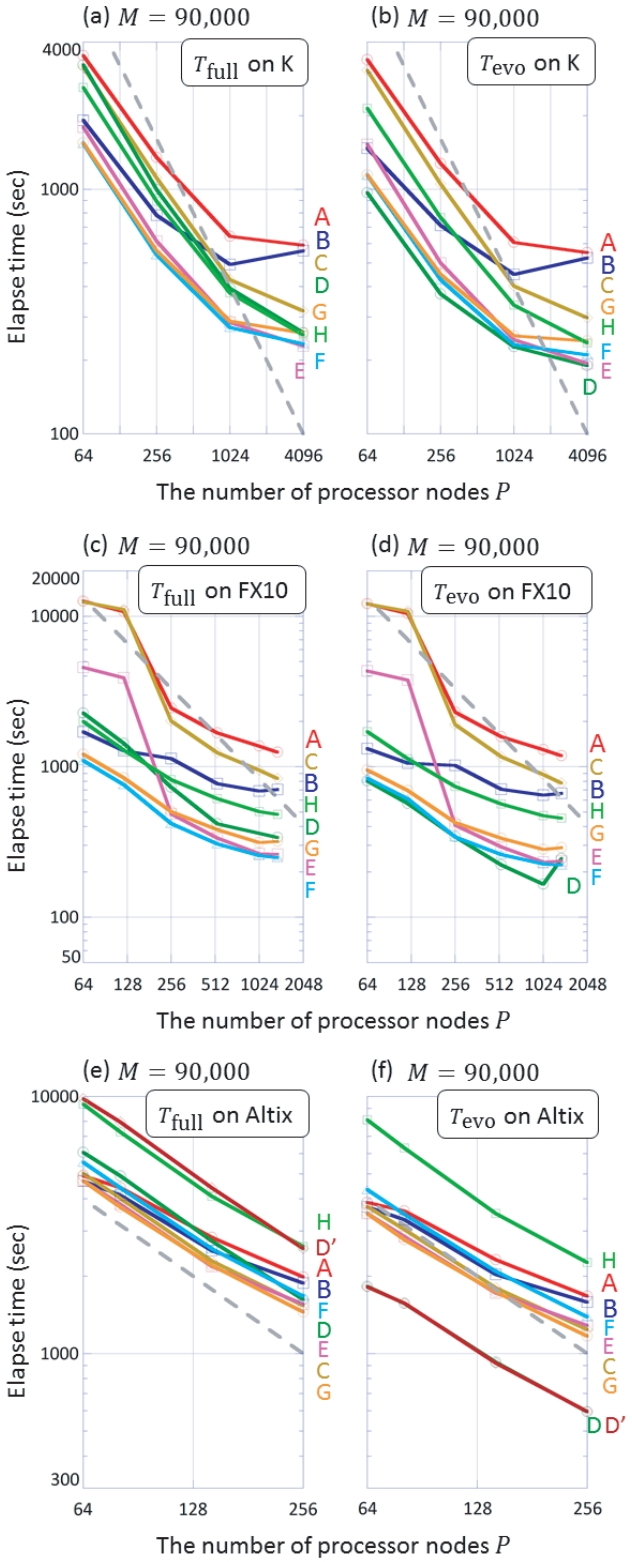

Fig. 5 Benchmark with the matrix size of M = 90,000, (I) on the K computer for the (a) full (eigenpair) and (b) eigenvalue-only calculation, (II) on FX10 for the (c) full and (d) eigenvalue-only calculation, (III) on Altix for the (e) full and (f) eigenvalue-only calculation. The ideal speedup in parallelism is drawn as a dashed gray line.

[*8] We observed on Altix that the 'ELPA2' and 'ELPA2′' SEP solver required non-blocking communication requests beyond the default limit number of $N_{\rm MPI\_MAX}$ = 16,384 and the job stopped with an MPI error message. Then we increased the limit number to $N_{\rm MPI\_MAX}$ = 1,048,576, the possible maximum of the machine by the environment variable 'MPI_REQUEST_MAX' and the calculations were completed.

[*9] One should remember that supercomputers are usually shared by many researchers who run many calculations in similar problem sizes successively and/or simultaneously.

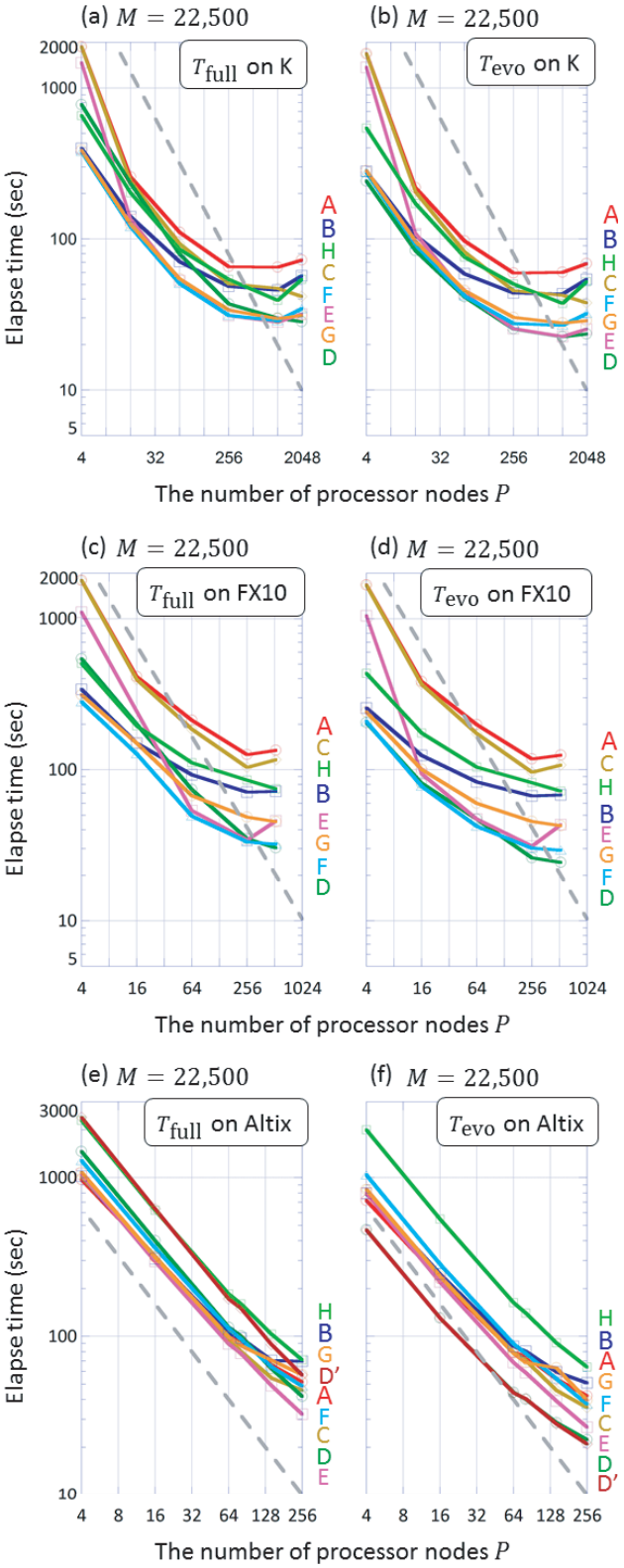

Fig. 6 Benchmark with the matrix size of M = 22,500, (I) on the K computer for the (a) full (eigenpair) and (b) eigenvalue-only calculation, (II) on FX10 for the (c) full and (d) eigenvalue-only calculation, (III) on Altix for the (e) full and (f) eigenvalue-only calculation. The ideal speedup in parallelism is drawn as a dashed gray line.

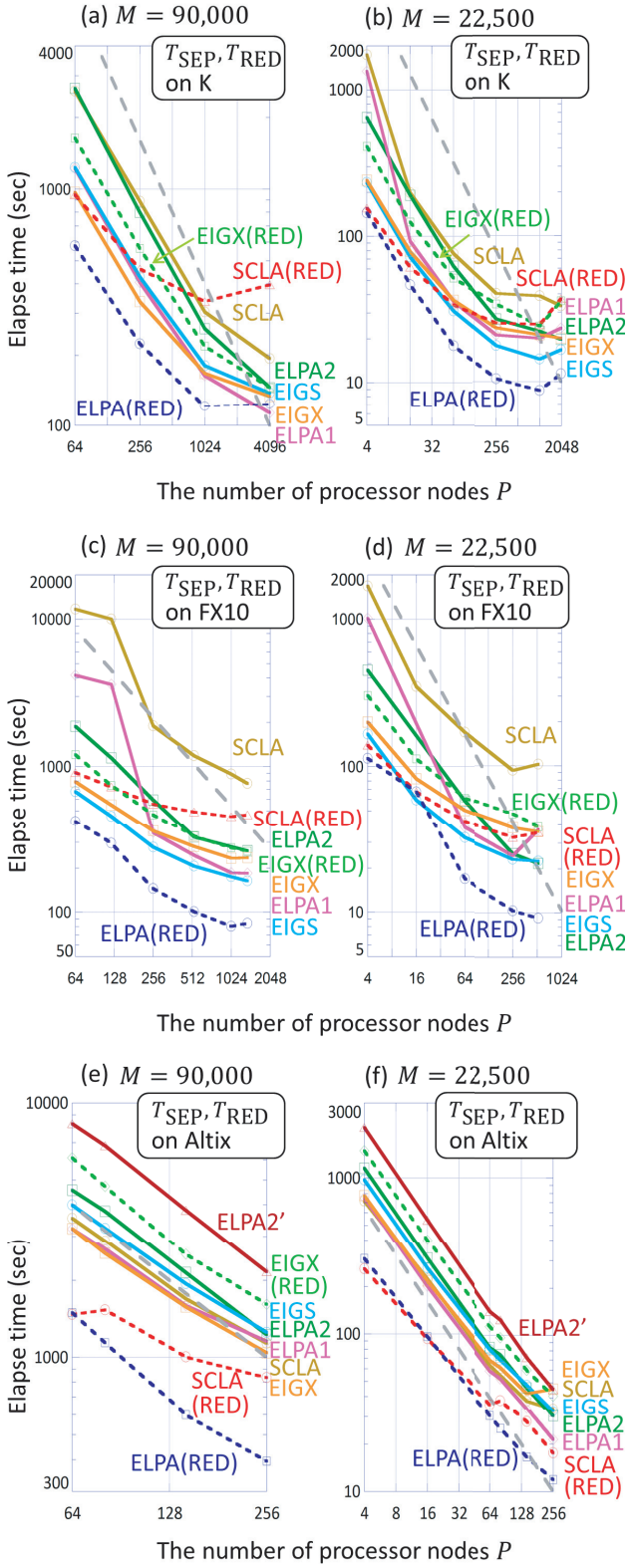

Fig. 7 Decomposition analysis of the elapse time into those of the SEP solver and the reducer (I) on the K computer with (a) M = 90,000 and (b) M = 22,500, (II) on FX10 with (c) M = 90,000 and (d) M = 22,500, (III) on Altix with (c) M = 90,000 and (d) M = 22,500. The routines for the reducers is labeled by '(RED)'. The 'ELPA2'' SEP solver is that without the SSE optimized routine. The ideal speedup in parallelism is drawn as a dashed gray line.

est elapse time in the full calculation appears among the workflows with the ELPA style reducer (the workflows D, E, F, and G) and that in the eigenvalue-only calculation appears with the workflow D. The above features are consistent with the results in the previous subsection. (II) Unlike the result in the previous subsection, the speed up is sometimes saturated. An example is observed in Fig. 6 (a), in the full calculation with M = 22,500 on

the K computer, because the elapse time in the workflow F gives a minimum as the function of P at P = 1,024. The decomposition analysis of Fig. 7 (b) indicates that the saturation occur both for the SEP solver and the reducer, which implies that the improvement both on the SEP solver and the reducer is desirable. The saturated cases are marked in Table 2 with the label of '[S]' [*10]. (III) Finally, the SSE-optimized routine in the workflow D is compared with the generic routine in the workflow D' in the case of M = 90,000 on Altix with P = 256. The SSE-optimized routine is prepared only in the backward transformation process. Since the process with the SSE-optimized routine or the generic one gives the elapse time of T_{BACK} = 929 sec or T_{BACK} = 1,872 sec, respectively, the process is accelerated with the SSE-optimized routine by 1,872 sec/929 sec ≈ 2.02. As shown in Table 2, the full calculation is accelerated with the SSE-optimized routine by 2,621 sec/1,621 sec ≈ 1.62.

4.3 Benchmark for a Million Dimensional Matrix

Finally, the benchmark for a million dimensional matrix is discussed. A press release at 2013 [24] reported, as a world record, a benchmark of a million dimensional SEP carried out by EigenExa, in approximately one hour, on the full (82,944) nodes of the K computer. An eigenvalue problem with a million dimensional matrix ($M = 10^6$) seems to be the practical limitation of the present supercomputer, owing to the $O(M^3)$ operation cost.

We calculated a million dimensional GEP in Dec. 2014 on the full (4,800) nodes of Oakleaf-FX [*11]. Since our computational resource was limited, only one calculation was carried out with the workflow G, because it gives the best data among those with M = 430,080 in Table 2. The calculation finished in approximately a half day, as shown in Table 2 ($T_{\rm full}$ = 39,919 sec and $T_{\rm evo}$ = 35,103 sec). The elapse time of the reducer ($T_{\rm RED} = T_{\rm full} - T_{\rm SEP}$ = 15,179 sec) is smaller than but comparable to that of the SEP solver ($T_{\rm SEP}$ = 24,740). The benchmark proved that the present code qualifies as a software applicable to massively parallel computation with up to a million dimensional matrix.

5. Discussions

5.1 Preparation of Initial Distributed Data

In the benchmark, the initial procedures including file reading are carried out for the preparation of distributed data. Its elapse time is always small and is ignored in the previous section [*12]. These procedures, however, may consume significant elapse times, when the present solver is used as a middleware with real applications. The discussions on such cases are beyond the present scope, since they depend on the program structure of the real applications. Here, several comments are added for real application developers: in general, the matrix data cost is, at most, $O(M^2)$ and the operation cost is $O(M^3)$ in the dense-matrix solvers and one should consider a balance between them. In the case of M = 430,080, for example, the required memory size for all the matrix elements is $8\,\mathrm{B} \times M^2 \approx 1.5\,\mathrm{TB}$, which can not be stored on a node of the K computer. Therefore, the data should be always distributed. In our real application (ELSES), the initial distributed data is prepared, when only the required elements are generated and stored on each node.

Table 4 The elapse times for data conversion; '(b → 1)', '(1 → b)' and '$T_{\rm RED}$' are the times in seconds for, the conversion process from block cyclic into cyclic distributions, the inverse process and the whole reducer procedure, respectively. The saturated data of $T_{\rm RED}$ are labelled by '[S]'. The 'ratio' is ((b → 1) + (1 → b))/$T_{\rm RED}$.

Size M	Machine(P)	(b → 1)	(1 → b)	$T_{\rm RED}$	ratio[%]
1,008,000	FX10(4,800)	51.4	51.7	8,208	1.26
430,080	K(10,000)	13.4	6.48	1,261	1.58
90,000	K(4,096)	6.89	0.797	124[S]	6.21
	FX10(1,369)	1.89	0.973	84.0[S]	3.41
	Altix(256)	2.01	2.02	394	1.02
22,500	K(2,025)	0.571	0.610	11.3[S]	10.4
	FX10(529)	0.328	0.176	9.20	5.48
	Altix(256)	0.120	0.279	11.9	3.35

5.2 Data Conversion Overhead

As explained in Section 3.1, several workflows require data conversion processes between distributed data formats, since ScaLAPACK and ELPA use block cyclic distribution with a given block size $n_{\rm block}(> 1)$ and EigenExa uses cyclic distribution ($n_{\rm block} \equiv 1$). In the present benchmark, the block size $n_{\rm block}$ in ScaLAPACK and ELPA was set to be $n_{\rm block}$ = 128, a typical value. Consequently, the workflows B, F, G require data conversion processes. In the present paper, the elapse time of the conversion procedures is included in the reducer part ($T_{\rm red}$).

Table 4 shows the elapse time for the data conversion. The elapse times are shown in the cases with the maximum numbers of used nodes ($P = P_{\rm max}$) among the present benchmark. Two data conversion procedures are required. One is the conversion from the block cyclic distribution into the cyclic distribution, shown as '(b → 1)' in Table 4 and the other is the inverse process shown as '(1 → b)'. The two procedures are carried out, commonly, by the 'pdgemr2d' routine in ScaLAPACK.

Table 4 indicates that the overhead of the data conversion procedures is always small and is not the origin of the saturation. In general, the conversion requires an $O(M^2)$ operation cost, while the calculation in a dense-matrix solver requires an $O(M^3)$ operation cost. The fact implies the general efficiency of hybrid solvers, at least, among dense-matrix solvers.

5.3 Decomposition Analysis of the Reducer

The decomposition analysis of the ELPA-style reducer is focused on, since the ELPA-style reducer is fastest among the three libraries. Figure 4 (c) shows the case on the K computer with M = 430,080. The elapse times of the subprocedures of the ELPA-style reducer are plotted; 'ELPA(R_1)' is the Cholesky factorization of Eq. (4), 'ELPA(R_2)' is the explicit calculation of the inversion $R = U^{-1}$ of the Cholesky factor U, 'ELPA(R_3)' and 'ELPA(R_4)' are the successive matrix multiplication of Eq. (5)

[*10] No saturation is found on Altix, unlike on the K computer and FX10, partially because the maximum number of used nodes ($P_{\rm max}$ = 256) is smaller.

[*11] We used FX10 not the K computer, because FX10 is in a newer architecture with a lower B/F value and the result on FX10 is speculated to be closer to that on the next-generation (exa-scale) machine.

[*12] In the case of the workflow G on the K computer with M = 430,080 and P = 10,000, for example, the elapse time of the initial procedures is $T_{\rm ini}$ = 123 sec and is much smaller than that of the total computation ($T_{\rm tot}$ = 2,734 sec. See Table 2). It is noteworthy that the present matrices are sparse, as explained in Section 2.

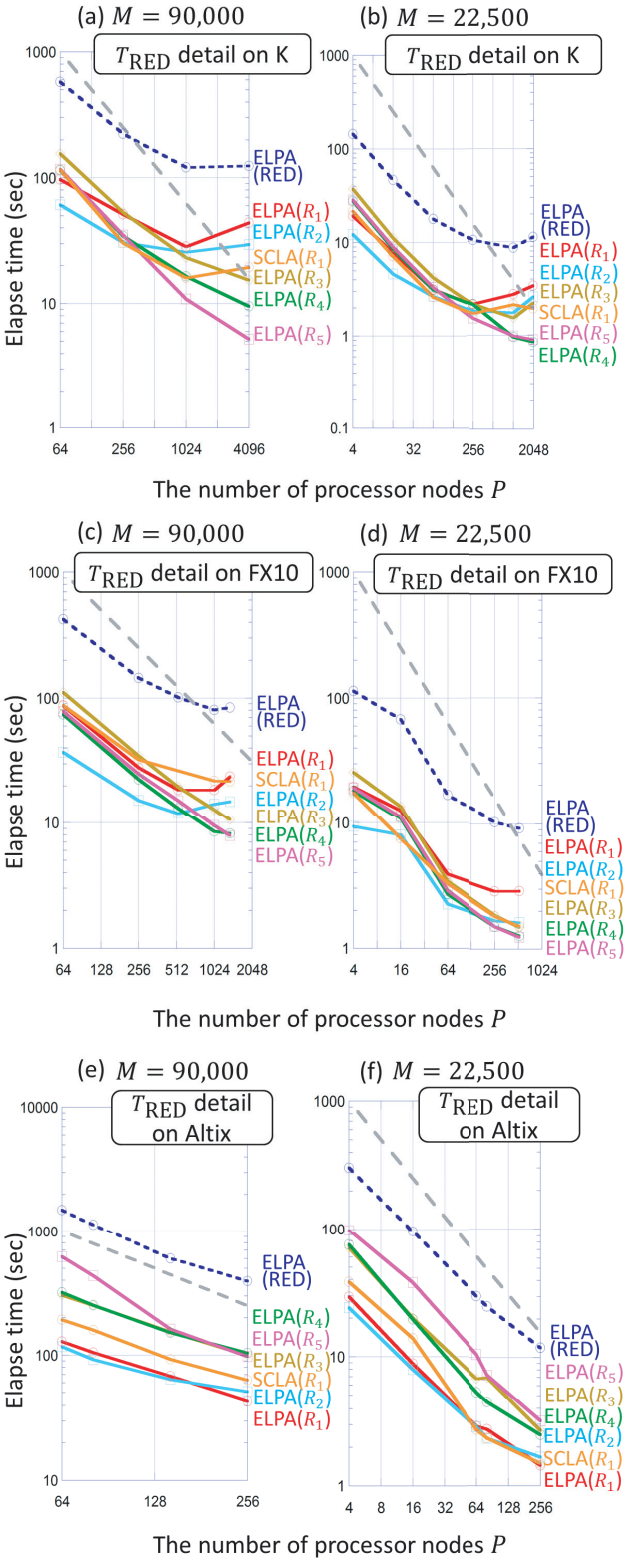

Fig. 8 Decomposition analysis of the elapse time of subprocedures of the ELPA style reducer and the Cholesky factorization in the ScaLAPACK style reducer (I) on the K computer with (a) M = 90,000 and (b) M = 22,500, (II) on FX10 with (c) M = 90,000 and (d) M = 22,500, (III) on Altix with (c) M = 90,000 and (d) M = 22,500. The ideal speedup in parallelism is drawn as a dashed gray line.

and 'ELPA(R_5)' is the backward transformation of eigenvectors by matrix multiplication of Eq. (6). The elapse times of the Cholesky factorization in the ScaLAPACK style reducer is also plotted as 'SCLA(R_1)' as a reference data. The same decomposition analysis is also carried out for other cases, as shown in **Fig. 8**. One can observe that the Cholesky factorization of the ELPA-style reducer does not scale and sometimes is slower than that of the ScaLAPACK reducer. In particular, the saturation of the ELPA-style reducer is caused by that of the Cholesky factorization in Fig. 8 (a)(b)(c).

The above observation implies that the reducer can be a serious bottleneck in the next-generation (exa-scale) supercomputers, though not in the present benchmark. One possible strategy is the improvement of the Cholesky factorization for better scalability and another is the development of a reducer without the Cholesky factorization, as in the EigenExa-style reducer.

6. Summary and Future Outlook

In summary, hybrid GEP solvers were constructed between the three parallel dense-matrix solver libraries of ScaLAPACK, ELPA and EigenExa. The benchmark was carried out with up to a million dimensional matrix on the K computer and other supercomputers. The hybrid solvers with ELPA and EigenExa give better benchmark results than the conventional ScaLAPACK library. The code was developed as a middleware and a mini-application and will appear online. Several issues are discussed. In particular, the decomposition analysis of the elapse time reveals a potential bottleneck on next-generation (exa-scale) supercomputers, which indicates the guidance for future development of the algorithms and the codes.

As a future outlook, the present code for the hybrid solvers is planned to be extended by introducing the solvers with different mathematical foundations. A candidate is the parallel block Jacobi solver [25], [26]. Since the solver is applicable only to standard eigenvalue problems, the hybrid solver enables us to use the solver in generalized eigenvalue problems.

Acknowledgments The authors thank Toshiyuki Imamura at the RIKEN Advanced Institute of Computational Science (AICS) and Takeshi Fukaya at Hokkaido University for fruitful discussions on EigenExa. The authors also thank Yusaku Yamamoto at The University of Electro-Communications on the parallel block Jacobi solver. This research is partially supported by Grant-in-Aid for Scientific Research (KAKENHI Nos. 25104718 and 26400318) from the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan. The K computer of RIKEN was used in the research projects of hp140069, hp140218, hp150144. The supercomputer Oakleaf-FX of the University of Tokyo was used in the research project of 14-NA04 in 'Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures' in Japan, in the 'Large-scale HPC Challenge' Project, Information Technology Center, The University of Tokyo and Initiative on Promotion of Supercomputing for Young or Women Researchers, Supercomputing Division, Information Technology Center, The University of Tokyo. We also used the supercomputer SGI altix ICE 8400EX at the Institute for Solid State Physics of the University of Tokyo and the supercomputers at the Research Center for Computational Science, Okazaki.

References

[1] Blackford, L.S. et al.: ScaLAPACK Users' Guide, Society for Industrial and Applied Mathematics, Philadelphia (1997).
[2] available from ⟨http://www.netlib.org/scalapack/⟩.
[3] Marek, A. et al.: The ELPA Library – Scalable Parallel Eigenvalue Solutions for Electronic Structure Theory and Computational Science, *J. Phys. Condens. Matter* **26**, 213201 (2014).
[4] Auckenthaler, T. et al.: Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations, *Parallel Computing*, Vol.37, Issue 12, pp.783–794 (2011).
[5] available from ⟨http://elpa.rzg.mpg.de/⟩.
[6] Imamura, T.: The EigenExa Library – High Performance & Scalable Direct Eigensolver for Large-Scale Computational Science, *ISC 2014*, Leipzig, Germany (2014).
[7] Imamura, T. et al.: EigenExa: high performance dense eigensolver, present and future, *8th International Workshop on Parallel Matrix Algorithms and Applications* (*PMAA14*), Lugano, Switzerland (2014).
[8] Imamura, T., Yamada, S. and Yoshida, M.: Development of a high-performance eigensolver on a peta-scale next-generation supercomputer system, *Prog. Nucl. Sci. Technol.*, Vol.2, pp.643–650 (2011).
[9] available from ⟨http://www.aics.riken.jp/labs/lpnctrt/index_e.html⟩.
[10] Hoshi, T., Yamazaki, K. and Akiyama, Y.: Novel Linear Algebraic Theory and One-Hundred-Million-Atom Electronic Structure Calculation on The K Computer, *JPS Conf. Proc.* **1**, 016004 (2014).
[11] Hoshi, T. et al.: Novel linear algebraic theory and one-hundred-million-atom quantum material simulations on the K computer, *PoS*(*IWCSE2013*)065 (2014).
[12] available from ⟨http://www.elses.jp/⟩.
[13] Hoshi, T. et al.: An order-*N* electronic structure theory with generalized eigenvalue equations and its application to a ten-million-atom system, *J. Phys. Condens. Matter* **21**, 165502 (2012).
[14] Sogabe, T., Hoshi, T., Zhang, S.L. and Fujiwara, T.: Solution of generalized shifted linear systems with complex symmetric matrices, *J. Comput. Phys.* 231, pp.5669–5684 (2012).
[15] available from ⟨http://www.elses.jp/matrix/⟩.
[16] Hoshi, T. et al.: Ten-million-atom electronic structure calculations on the K computer with a massively parallel order-*N* theory, *J. Phys. Soc. Jpn.* 82, 023710 (2013).
[17] Calzaferri, G. and Rytz, R.: J. Phys. Chem. 100, 11122 (1996).
[18] Cerdá, J. and Soria, F.: Phys. Rev. B **61**, pp.7965–7971, (2000).
[19] available from ⟨http://www.aics.riken.jp/labs/lpnctrt/KMATH_EIGEN_GEV_e.html⟩.
[20] Golub, G.H. and Van Loan, C.F.: *Matrix Computations*, 4th Ed., Johns Hopkins University Press, Baltimore, MD (2013).
[21] Tisseur, F. and Dongarra, J.: Parallelizing the divide and conquer algorithm for the symmetric tridiagonal eigenvalue problem on distributed memory architectures, *SIAM J. Sci. Comput.*, Vol.20, Issue 6, pp.2223–2236 (1999).
[22] Poulson, J., v. d. Geijn, R. and Bennighof, J.: Parallel algorithms for reducing the generalized hermitian-definite eigenvalue problem, FLAME Working Note #56, The University of Texas at Austin, Department of Computer Science, Tech. Rep. TR-11-05 (2011).
[23] Sears, M.P., Stanley, K. and Henry, G.: Application of a High Performance Parallel Eigensolver to Electronic Structure Calculations, *Proc. 1998 ACM/IEEE conference on Supercomputing* (*SC '98*), Orlando, FL (1998).
[24] RIKEN, Press Release at 5. Dec. 2013 (in Japanese), available from ⟨http://www.riken.jp/pr/press/2013/20131205_1/⟩.
[25] Takahashi, Y., Hirota, Y. and Yamamoto, Y.: Performance of the block Jacobi method for the symmetric eigenvalue problem on a modern massively parallel computer, *Proc. ALGORITMY 2012*, pp.151–160 (2012).
[26] Yamamoto, Y., Zhang, L. and Kudo, S.: Convergence analysis of the parallel classical block Jacobi method for the symmetric eigenvalue problem, *JSIAM Letters*, Vol.6, pp.57–60 (2014).

Hiroto Imachi was born in 1988. He received his M.E. from the University of Tokyo in 2013. He is currently a doctoral student at Tottori University. His research interest is massively parallel computational algorithms and their application to quantum material simulations.

Takeo Hoshi was born in 1970. He received his M.E. and Ph.D. from the University of Tokyo in 1995 and 2003, respectively. He became a research associate of the University of Tokyo in 1995 and an associate professor of Tottori University in 2006. His research interest is the disciplinary research on quantum material simulations and massively parallel computational algorithms.