Parallel Iterative Solvers based on Domain Decomposition with Robust Preconditioners for Ill-conditioned Problems

Kengo Nakajima

Information Technology Center, The University of Tokyo
2-11-16 Yayoi, Bunkyo-ku, Tokyo 113-8658, Japan
nakajima@cc.u-tokyo.ac.jp

***Abstract*— Performance and robustness of preconditioned parallel iterative solvers based on domain decomposition for ill-conditioned problems are evaluated in the present work. The preconditioning method is based on BILUT(p,d,t) originally proposed by the author, and two types of domain decomposition procedures, LBJ (Localized Block Jacobi) and HID (Hierarchical Interface Decomposition), are considered. Developed methods are applied to Hetero3D code, which is a parallel finite-element benchmark program for solid mechanics problems, and the code provided excellent scalability and robustness up to 240 nodes (3,840 cores) of Fujitsu PRIMEHPC FX10 (Oakleaf-FX), Information Technology Center, the University of Tokyo. Generally, HID provides better performance and robustness than LBJ for a wide range of values of parameters.**

Keywords—Parallel Iterative Solvers; Preconditioning Methods; Ill-conditioned Problems

I. Introduction

Incomplete LU factorization with threshold (ILUT) [1,2] is a widely-used preconditioning technique for solving various types of problems with ill-conditioned matrices. ILUT(*p,d,t*) [3,4], proposed by the author, is a more practical one, where maximum fill-level *p* is specified before factorization, and *d* and *t* are parameters for dropping tolerance before/after factorization. Figure 1 describes the procedures of ILUT(*p,d,t*) method. The process (b) in Fig.1 can be substituted by other factorization methods or more powerful direct linear solvers, such as *MUMPS* [5], *SuperLU* [6] and etc.

In the present work, parallel preconditioned iterative solvers based on ILUT(*p,d,t*) method were developed, and were implemented to *Hetero3D* code. Hetero3D is a benchmark code for parallel finite-element method (FEM) developed under JST-ANR FP3C project (Collaborative Project between Japan and France on Framework and Programming for Post Petascale Computing) [7]. It is widely known that convergence of parallel preconditioned iterative solvers is strongly affected by methods for domain decomposition, especially for ill-conditioned problems [3,4]. In the present work, two types of domain decomposition methods, *LBJ* (Localized Block Jacobi) [3,4] and *HID* (Hierarchical Interface Decomposition) [3,4], are applied.

Large-scale ill-conditioned matrices derived from *Hetero3D* code were solved using up to 240 nodes (3,840 cores) of Fujitsu PRIMEHPC FX10 (Oakleaf-FX), Information Technology Center, the University of Tokyo [8]. Finally, performance and robustness of preconditioned parallel iterative solvers based on domain decomposition for ill-conditioned problems are evaluated.

The rest of this paper is organized as follows. In section II, we outline the target application and preconditioned iterative solvers. In section III an overview of the target hardware is provided. In sections IV and V, the results of the computations are described, while related work and final remarks are offered in section VI and VII.

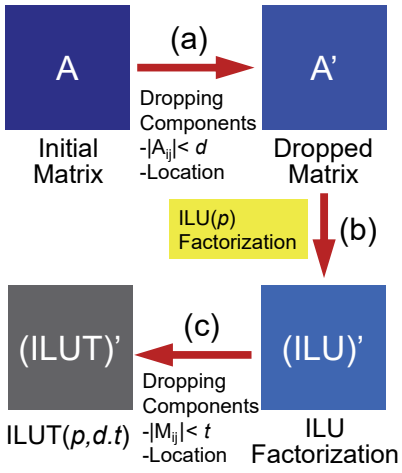


Fig.1 Procedures of ILUT(*p,d,t*) preconditioning, *p*: level of fill-in's, *d*: dropping tolerance before factorization, *t*: dropping tolerance after factorization

II. Target Application and Linear Solvers

A. Hetero3D

The Hetero3D solves linear elasticity problems in simple cube geometries of media with heterogeneous material properties (Fig.2(a)) using parallel FEM. Tri-linear hexahedral (cubic) elements are used for the discretization, and a heterogeneous distribution of Young's modulus in each element is calculated by a sequential Gauss algorithm, which is widely used in the area of geostatistics [9]. The boundary conditions are described in Fig.2(b).

The Hetero3D code is based on the framework for parallel FEM procedures of GeoFEM [10], and the GeoFEM's local

data structure is applied. The Hetero3D generates parallel distributed local meshes and coefficient matrices in parallel manner using MPI. In the Hetero3D, total number of vertices in each direction (N_x, N_y, N_z), and number of partitions in each direction (P_x,P_y,P_z) are specified before computation. Number of total MPI processes is equal to $P_x \times P_y \times P_z$, and each MPI process has $(N_x/P_x)\times(N_y/P_y)\times(N_z/P_z)$ vertices. Spatial distribution of Young's modulus is given by an external file, which includes information for heterogeneity for the field of 128^3 cube geometry. If N_x (or N_y or N_z) is larger than 128, distribution of these 128^3 cubes is repeated periodically in each direction.

Only flat MPI parallel programming model can be applied to the current version of the Hetero3D.

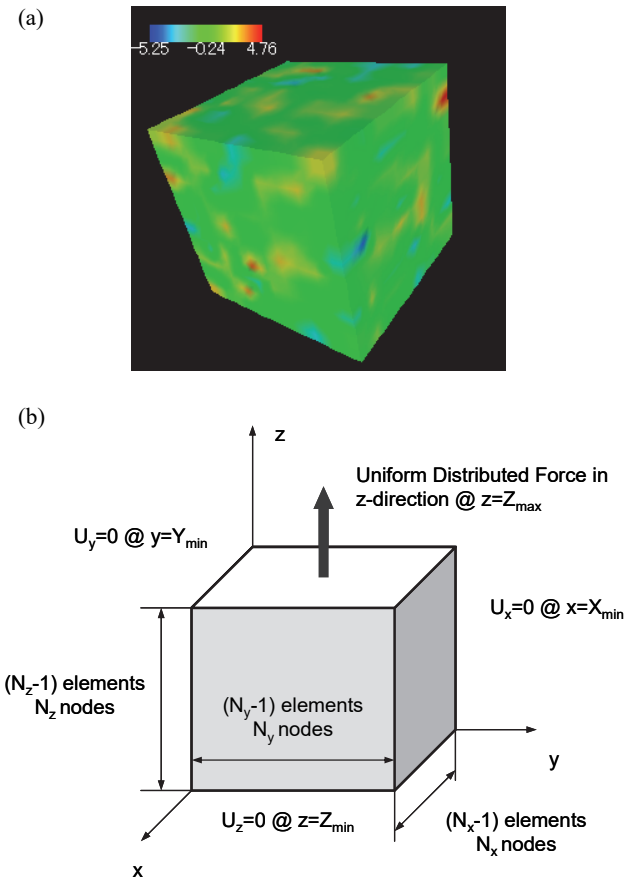


Fig.2 Simple cube geometries with heterogeneity as domains for 3D linear elasticity problems (a) distribution of Young's modulus, (b) boundary conditions

B. Preconditioned Iterative Linear Solvers

In the present work, derived matrices are solved by GPBi-CG (Generalized Product-type methods based on Bi-CG) [11] iterative method, and ILUT(p,d,t) preconditioning is applied, where p, d and t are defined as follows:

- p: The maximum level of the fill-in (=p) for Incomplete LU (ILU) factorization[3,4]
- d: Threshold parameter for dropping tolerance *before* factorization, A_{ij} (each component of initial coefficient matrix A) is set to 0 if A_{ij} is smaller than d (Fig.1)
- t: Threshold parameter for dropping tolerance *after* factorization, M_{ij} (each component of factorized matrix M by BILU) is set to 0 if M_{ij} is smaller than d (Fig.1)

ILUT(p,d,t) is more practical than traditional ILUT, because it specifies the maximum level of the fill-in (=p) for Incomplete LU (ILU) factorization before computation. Thus, both of memory and computational amount are much more efficiently saved, compared to that of traditional ILUT. The *Hetero3D* solves 3D linear elasticity problems, and each vertex has three degrees-of-freedom (DOF), (u,v,w) where each component denotes displacement in each of x-, y-, and z-directions. These three components are tightly coupled, therefore these are processed simultaneously with 3-by-3 (3×3) block-wise manner in matrix operations [3,4]. This block-wise manner provides efficiency in the computation of matrix operations in the Hetero3D. ILUT(p,d,t) preconditoing is also done by this block-wise manner, therefore ILUT(p,d,t) is called BILUT(p,d,t) (Block ILUT). BILUT(p,d,t) preconditioning described in Fig.1 is applied to each local matrix on each MPI process.

It is widely known that convergence of parallel preconditioned iterative solvers is strongly affected by methods for domain decomposition, especially for ill-conditioned problems [3,4]. In the present work, two types of domain decomposition methods, *LBJ* (Localized Block Jacobi) [3,4] and *HID* (Hierarchical Interface Decomposition) [3,4], are applied

C. Localized Block Jacobi (LBJ)

LBJ-based preconditioners are widely used for parallel iterative solvers. They provide excellent parallel performance for well-defined problems, but it is not robust for ill-conditioned problems with many processors, because it ignores the global effect of external nodes in other domains. The generally used remedy is the extension of overlapped elements between domains [3,4,12] (Fig.3).

In the present work, various configurations of thickness of overlapped zones between distributed local meshes for parallel FEM are evaluated for LBJ, where LBJ-x means LBJ with x layers of overlapped elements.

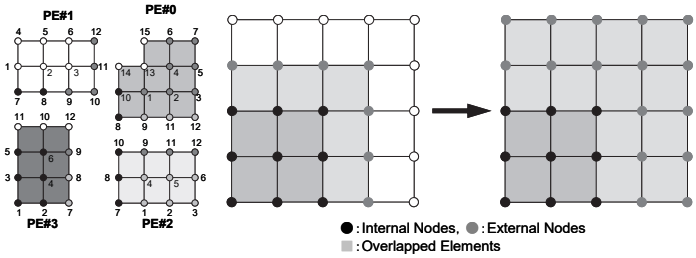
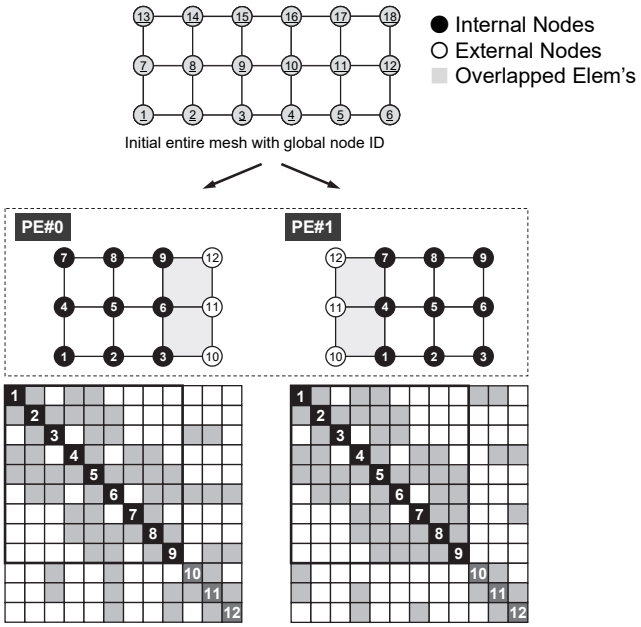
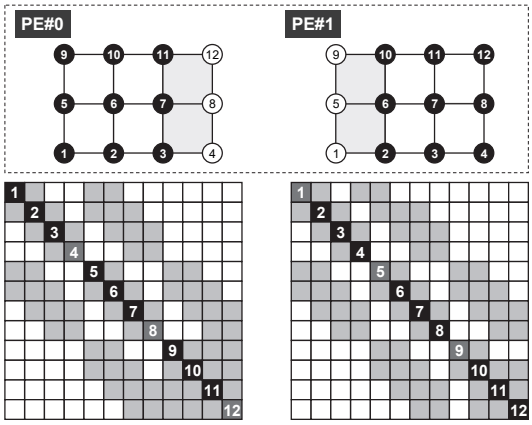


Fig.3 Extension of depth of overlapping [3,4]



(a)Internal-External numbering



(b) *RCM-Entire* numbering

Fig.4 Distributed meshes and coefficient matrices for parallel FEM operations

It is widely known that the reordering of vertices strongly affects the convergence of iterative solvers with ILU-type preconditioners [12]. Vertices (nodes) in each local mesh data of the GeoFEM data structure are classified into the following *three* categories from the viewpoint of message passing [10,12]:

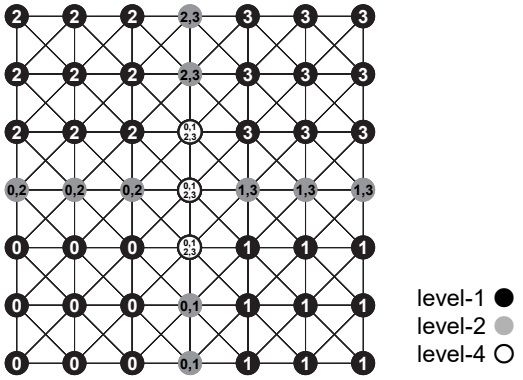
- Internal nodes (originally assigned to the domain)
- External nodes (forming an element in the domain, but are from external domains)
- Boundary nodes (external nodes of other domains)

Figure 3(a) describes a very simple example, where an initial entire mesh comprising 18 nodes and 10 quadrilateral elements is partitioned into two domains. Local numbering starts from the internal nodes. The external nodes are numbered after all the internal nodes have been numbered, as shown in Fig.3(a). According to this numbering method (*internal-external* numbering), the bandwidth of local sparse coefficient matrices including the external nodes is relatively large, as shown in Fig.3(a). Coefficient matrices with larger bandwidth usually provide slower convergence for iterative solvers with ILU-type preconditioning methods [1]. As long as fully localized block Jacobi-type preconditioning methods in [10] are adopted, this effect of bandwidth is rather smaller. But if overlapping among domain is applied, this effect may be more significant.

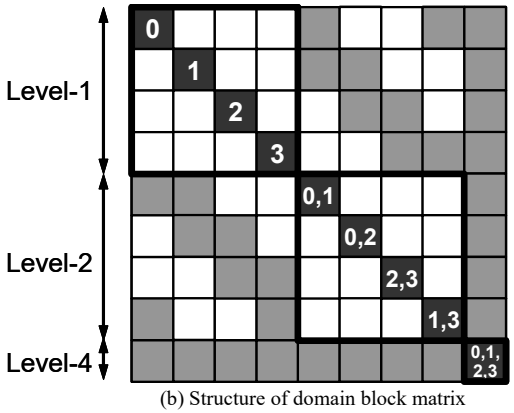
In [12], *RCM-entire* numbering was introduced, where both the internal and external nodes in each domain are reordered according to Reverse Cuthill-Mckee (RCM) method [1]. Figure 3(b) shows local data meshes obtained through this *RCM-entire* numbering, and corresponding local coefficient matrices. It may be noted that both of the bandwidth of local coefficient matrices and number of fill-in's of factorized matrices are smaller than that of original matrices in Fig.3(a). In the present work, *RCM-entire* numbering is applied.

D. Hierarchical Interface Decomposition (HID)

Hierarchical Interface Decomposition (HID) provides a method of domain decomposition with robustness and scalability for parallel preconditioners based on the ILU and IC (Incomplete Cholesky) factorizations, and it is expected to be more robust than LBJ-based ones.



(a) Domain decomposition (*connectors* and *keys*)



(b) Structure of domain block matrix

Fig.5 Domain/block decomposition of the matrix according to the HID reordering

The HID process starts with a partitioning of the graph into sub-graphs, where there is no overlapping between sub-graphs.

These sub-graphs are called *connecters with level-1* (C^1). Because each C^1 connector corresponds to each domain in parallel computation with domain decomposition, C^1 connectors are called *sub-domains*. The "levels" are defined recursively from this partitioning, with each level. Each vertex group of a given level is a *separator* for vertex groups of a *lower* level. In [8], the concepts of *connectors* of different *levels* (C^k) are introduced, where C^k connectors are adjacent to k *sub-domains*. Figure 4(a) shows the example of the partition of a 9-point grid into 4 domains. Note that different connectors of the same level are not connected directly, but are separated by connectors of higher levels. If the unknowns are reordered according to their level numbers, from the lowest to highest, the block structure of the reordered matrix is as shown in Fig.4(b). This block structure leads to a natural parallelism if ILU/IC factorization or forward/backward substitution processes are applied. Details of algorithms for the construction of independent connectors are provided in [13]. Thus, HID-based ILU/IC type preconditioners can consider the global effect in parallel computations, and are expected to be more robust than LBJ-based ones. In [12], original partitioner of GeoFEM for domain decomposition was modified so that it could create a distributed hierarchical data structure for HID.

```
do lev= 1, LEVELtot
  do i0= LEVELindex(lev-1)+1, LEVELindex(lev)
    i= OtoN(i0)
    SW1= WW(3*i-2,R); SW2= WW(3*i-1,R); SW3= WW(3*i  ,R)
    isL= INL(i-1)+1; ieL= INL(i)
    do j= isL, ieL
        k= IAL(j)
       X1= WW(3*k-2,R); X2= WW(3*k-1,R); X3= WW(3*k  ,R)
      SW1= SW1 - AL(9*j-8)*X1 - AL(9*j-7)*X2 - AL(9*j-6)*X3
      SW2= SW2 - AL(9*j-5)*X1 - AL(9*j-4)*X2 - AL(9*j-3)*X3
      SW3= SW3 - AL(9*j-2)*X1 - AL(9*j-1)*X2 - AL(9*j  )*X3
    enddo
    X1= SW1; X2= SW2; X3= SW3
    X2= X2 - ALU(9*i-5)*X1
    X3= X3 - ALU(9*i-2)*X1 - ALU(9*i-1)*X2
    X3= ALU(9*i  )*  X3
    X2= ALU(9*i-4)*( X2 - ALU(9*i-3)*X3 )
    X1= ALU(9*i-8)*( X1 - ALU(9*i-6)*X3 - ALU(9*i-7)*X2)
    WW(3*i-2,R)= X1; WW(3*i-1,R)= X2; WW(3*i  ,R)= X3
  enddo

  call SOLVER_SEND_RECV_3_LEV(lev,…):   Communications using
                                        Hierarchical Comm. Tables.
enddo
```

Fig.6 Forward substitution process of preconditioning written in Fortran and MPI, global communications using hierarchical communication tables occur at the end of the computation of each level

Figure 5 shows *forward substitution* process of preconditioning, written in Fortran and MPI. Global communications using hierarchical communication tables for HID procedures occur in the end of the computation at each level. HID is more expensive than LBJ due to these additional communications.

Although the author extended HID for robustness using thicker separators in [3], original implementation of HID in [13] is applied in this study.

III. Hardware Environment

"Fujitsu FX10" system at the University of Tokyo (Oakleaf-FX) [8] is Fujitsu's PRIMEHPC FX10 massively parallel supercomputer with a peak performance of 1.13 PFLOPS. Oakleaf-FX consists of 4,800 computing nodes of SPARC64™ IXfx processors with sixteen cores (1.848 GHz) (Fig.6) [14]. The SPARC64™ IXfx incorporates many features for HPC, which includes SPARC64™ V9 specification with HPC-ACE (Arithmetic Computational Extensions) and a hardware barrier for high-speed synchronization of on-chip cores [14]. Entire system consists of 76,800 cores and 154 TB memory.

Table 1: Summary of Specification of a Single Node of Fujitsu FX10

Core #/Node	16
Size of Memory/node (GB)	32
Peak Performance/node (GFLOPS)	236.5
Peak Memory Bandwidth/node (GB/sec)	85.3
STREAM/Triad Performance/node (GB/sec) [15]	64.7
B/F Rate	0.274

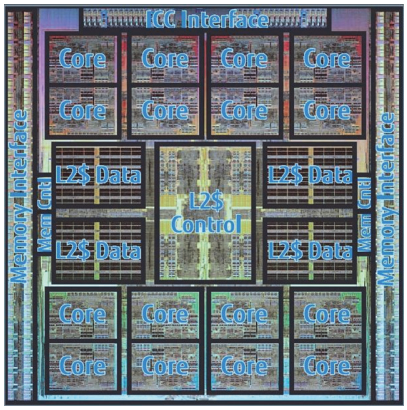


Fig.7 Fujitsu's SPARC64™ IXfx Processor [14]

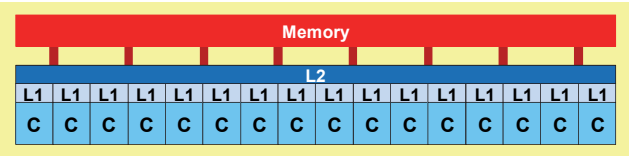


Fig.8 Overview of a computing node of the Fujitsu FX10 (C: core, L1/L2/L3: cache, Memory: main memory)

Each core has a 64 KB L1 instruction/data cache. A 12 MB L2 cache is shared by sixteen cores on each node. Nodes are connected via 6-dimensional mesh/torus interconnect, called "Tofu". Each SPARC64™ IXfx processor is connected to a dedicated interconnect controller (ICC) (Fig.6) [14].The ICC uses four Tofu network interface (TNI) to perform simultaneous 4-way transmission and 4-way reception. In the present work, up to 240 nodes of the system have been evaluated. Although users can specify the topology of network on Fujitsu FX10, this capability was not used in the current study. On the SPARC64™ IXfx, each of sixteen cores can access to memory in uniform manner (Fig.7).

IV. Preliminary Results

In this section, preliminary study has been conducted for choosing appropriate configurations for following issues:

- Optimum value of d in BILUT(p,d,t): dropping tolerance of BILUT(p,d,t) before factorization
- Optimum value of x of LBJ-x: thickness of overlapped zones between distributed local meshes for parallel FEM

- Numbering and reordering of distributed local meshes: *Internal-External* numbering or *RCM-Entire* numbering on LBJ cases

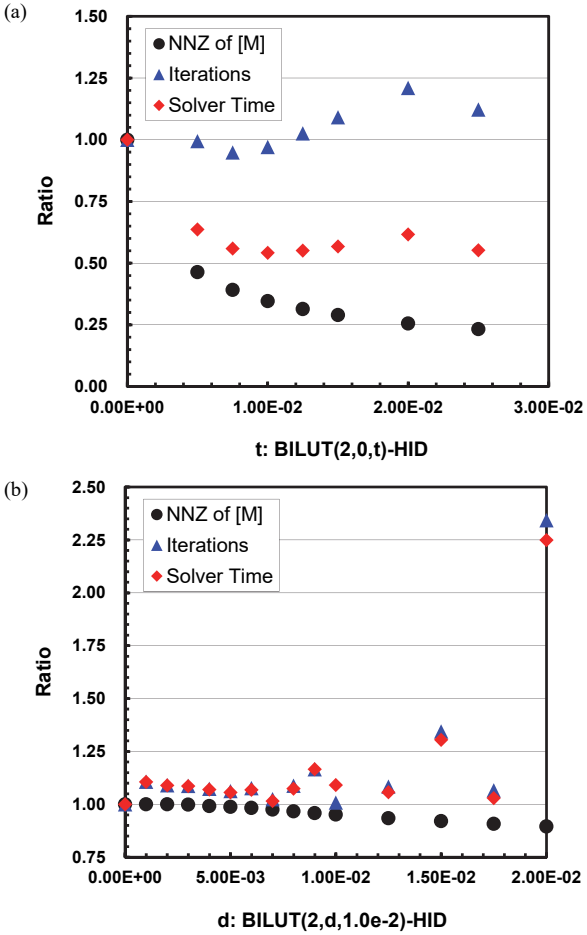


Fig.9 Effect of d and t on convergence and performance of BILUT(2,d,t)-HID-GPBi-CG with 240 nodes (3,840 cores) of the of the Oakleaf-FX, E_{max}=10^{-6}, E_{max}=10^{+6}, NNZ of [M]: Total number of non-zero components (NNZ) in the factorized matrices for BILUT(p,d,t) ([M]) (a) BILUT(2,0,t)-HID, results are normalized by those of BILUT(2,0,0)-HID, (b) BILUT(2,d,1.00×10^{-2})-HID, results are normalized by those of BILUT(2,0, 1.00×10^{-2})-HID

Target problem is a 3D linear elastic problem with 400×320×240 vertices (=3.072×10^7 vertices, 30,420,159 elements, 9.216×10^7 DOF). 240 nodes (3,840 cores) of the Oakleaf-FX system has been applied.

A. Effect of d and t in BILUT(p,d,t)

First of all, effect of parameters d and t in BILUT(p,d,t)-HID has been evaluated on a 3D linear elastic problem. The minimum and maximum values of Young's modulus (E_{min} and E_{max}) are 10^{-6} and 10^{+6}, respectively, with an average value of 1.0.

Figure 8(a) shows the effect of t on convergence and performance of BILUT(2,0,t)-HID, while Fig.8(b) shows that of d on convergence and performance of BILUT(2,d,1.00×10^{-2})-HID. "NNZ of [M]" in Fig.8(a) and Fig.8(b) means total number of non-zero components (NNZ) in the factorized matrices for BILUT(p,d,t) ([M]). "Iterations" means number of iterations until convergence, while "Solver Time" means elapsed computation time for linear solver, where time for factorization is not included.

In Fig.8(a), d is fixed as 0, and t is changed from 0.00 to 2.00×10^{-2}, and results are normalized by those of BILUT(2,0,0)-HID. Generally speaking, *NNZ* decreases, and number of iterations until convergence increases, if d increases. Because NNZ corresponds to computational cost for each iteration, product of NNZ and iterations will be the time for computation. In this case, t=1.00×10^{-2} provides the best performance from the view point of elapsed computation time. If t is optimum, Elapsed computation time for linear solver of BILUT(2,0,1.00×10^{-2})-HID is only 50% as large as that of BILUT(2,0,0)-HID without dropping.

In Fig.8(b), t is fixed as the optimum value in Fig.8(a) (=1.00×10^{-2}), and d is changed from 0.00 to 2.00×10^{-2}, and results are normalized by those of BILUT(2,0, 1.50×10^{-2})-HID. Effect of d is very small, if d is less than 1.50×10^{-2}, and convergence gets significantly worse if d is larger than 1.50×10^{-2}, where the important features of the original matrices are lost.

These results show that d is not an effective parameter on the improvement of convergence of BILUT(p,d,t)-HID-GPBi-CG, therefore we fix d as 0 in the present work. On the contrast, contribution of dropping after factorization on improvement is very large. Optimum value of t varies according to feature of problems, condition number, size of problems, and many other parameters. We will investigate the effect of t in the next section.

B. Effect of thickness of overlapped zones

Next, effect of thickness of overlapped zones between distributed local meshes for parallel FEM is evaluated for LBJ, where LBJ-x means LBJ with x layers of overlapped elements. The minimum and maximum values of Young's modulus are 10^{-6} and 10^{+6}, respectively, with an average value of 1.0. *RCM-Entire* numbering is applied to the numbering of distributed local meshes.

Table 2(a) and (b) show the results. In Table 2(a), values of d and t are set to zero in BILU(p,d,t), while Table 2(b) shows the results with optimum selection of t. Generally speaking, LBJ-2 provides the best elapsed time except BILUT(3,0, 2.50×10^{-2})-LBJ-2, although BILUT(3,0, 2.50×10^{-2})-LBJ-2 is the fastest in linear solver among 18 cases in Table 2. Therefore, we fix x of LBJ-x as 2 in the present work.

"Set-up" time in Table-2 means computation time of BILUT matrices, including factorization and dropping. Generally speaking, if the level of fill-in (p) and depth of overlapping increase, time for set-up increases. This is because the amount of computation increases according to these parameters. It is clear that larger level of fill-in's and depth of overlapping provide better convergence, but cost for computation for both of set-up and linear solver increases. Finding the optimum combination of parameters (p, d, t, and depth of overlapping (x of LBJ-x) etc.) is a very critical issue for efficiency and robustness.

Table 2: Effect of Thickness of Overlapped Zones of LBJ on convergence and performance of BILUT(*p,d,t*)-LBJ-*x*-GPBi-CG with 240 nodes (3,840 cores) of the of the Oakleaf-FX, $E_{max}=10^{-6}$, $E_{max}=10^{+6}$, *RCM-Entire* numbering (a) BILUT(*p,0,0*)-LBJ-*x*, (b) BILUT(*p,0,t*) -LBJ-*x*, with Optimum *t*

(a)

Preconditioner & Domain Decomposition	NNZ	Set-up (sec.)	Solver (sec.)	Total (sec.)	Iter's
BILUT(1,0,0)-LBJ-1	1.920×10^{10}	1.35	65.2	66.5	1916
BILUT(1,0,0)-LBJ-2	2.519×10^{10}	2.03	61.8	63.9	1288
BILUT(1,0,0)-LBJ-3	3.197×10^{10}	2.79	74.0	76.8	1367
BILUT(2,0,0)-LBJ-1	3.351×10^{10}	3.09	71.8	74.9	1339
BILUT(2,0,0)-LBJ-2	4.394×10^{10}	4.39	65.2	69.6	939
BILUT(2,0,0)-LBJ-3	5.631×10^{10}	5.95	83.6	89.6	1006
BILUT(3,0,0)-LBJ-1	6.468×10^{10}	9.34	105.2	114.6	1192
BILUT(3,0,0)-LBJ-2	8.523×10^{10}	12.7	98.4	111.1	823
BILUT(3,0,0)-LBJ-3	1.101×10^{11}	17.3	101.6	118.9	722

(b)

Preconditioner & Domain Decomposition	NNZ	Set-up (sec.)	Solver (sec.)	Total (sec.)	Iter's
BILUT(1,0,2.75 $\times10^{-2}$)-LBJ-1	7.755×10^{9}	1.36	45.0	46.3	1916
BILUT(1,0,2.75 $\times10^{-2}$)-LBJ-2	1.019×10^{10}	2.05	42.0	44.1	1383
BILUT(1,0,2.75 $\times10^{-2}$)-LBJ-3	1.285×10^{10}	2.81	54.2	57.0	1492
BILUT(2,0,1.00 $\times10^{-2}$)-LBJ-1	1.118×10^{10}	3.11	39.1	42.2	1422
BILUT(2,0,1.00 $\times10^{-2}$)-LBJ-2	1.487×10^{10}	4.41	37.1	41.5	1029
BILUT(2,0,1.00 $\times10^{-2}$)-LBJ-3	1.893×10^{10}	5.99	37.1	43.1	915
BILUT(3,0,2.50 $\times10^{-2}$)-LBJ-1	8.072×10^{9}	9.35	38.4	47.7	1526
BILUT(3,0,2.50 $\times10^{-2}$)-LBJ-2	1.063×10^{10}	12.7	35.5	48.3	1149
BILUT(3,0,2.50 $\times10^{-2}$)-LBJ-3	1.342×10^{10}	17.3	40.9	58.2	1180

C. *Effect of numbering for distributed local meshes*

Table 3 shows results of BILUT(3,0,2.50$\times10^{-2}$)-LBJ-2 on a 3D linear elastic problem, and compares effect of *Internal-External* and *RCM-Entire* numbering for distributed local meshes, described in Fig.3. *RCM-Entire* numbering provides much better convergence than *Internal-External*. Moreover set-up time is much more reduced. This is because of the feature of RCM-Entire numbering, which reduced both of bandwidth and fill-in's. In the present work, we apply *RCM-Entire* numbering for LBJ cases.

Table 3: Effect of numbering for distributed local meshes on BILUT(3,0,2.50$\times10^{-2}$)-LBJ-2-GPBi-CG with 240 nodes (3,840 cores) of the of the Oakleaf-FX, $E_{max}=10^{-6}$, $E_{max}=10^{+6}$

Numbering of distributed local meshes	NNZ	Set-up (sec.)	Solver (sec.)	Total (sec.)	Iter's
RCM-Entire	1.063×10^{10}	12.7	35.5	48.3	1149
Internal-External	1.102×10^{10}	20.2	49.6	69.8	1690

V. Results

A. *Effect of t in BILU(p,d,t)*

Effect of parameter *t* in BILUT(*p,d,t*)-HID and BILUT(*p,d,t*)-LBJ-*x* has been evaluated on a 3D linear elastic problem with 400×320×240 vertices (9.216×10^{7} DOF) using 240 nodes (3,840 cores) of the Oakleaf-FX system. The minimum and maximum values of Young's modulus (E_{min} and E_{max}) are 10^{-6} and 10^{+6}, respectively, with an average value of 1.0. Value of *d* is fixed as 0.00, and *x* of LBJ-*x* is set to 2 according to the results of the previous section.

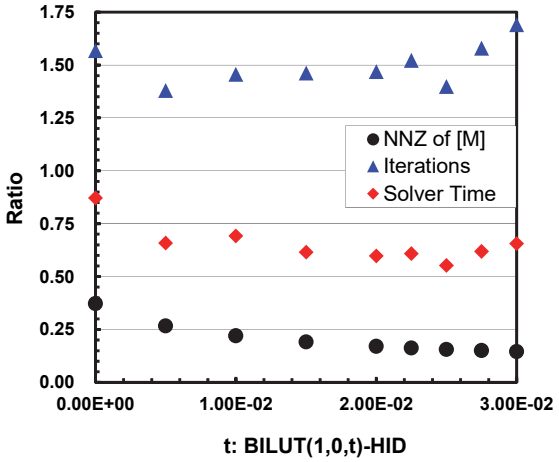


Fig.10 Effect of *t* on convergence and performance of BILUT(1,0,*t*)-HID-GPBi-CG with 240 nodes (3,840 cores) of the of the Oakleaf-FX, $E_{max}=10^{-6}$, $E_{max}=10^{+6}$, Normalized by results of BILUT(2,0,0)-LBJ-2

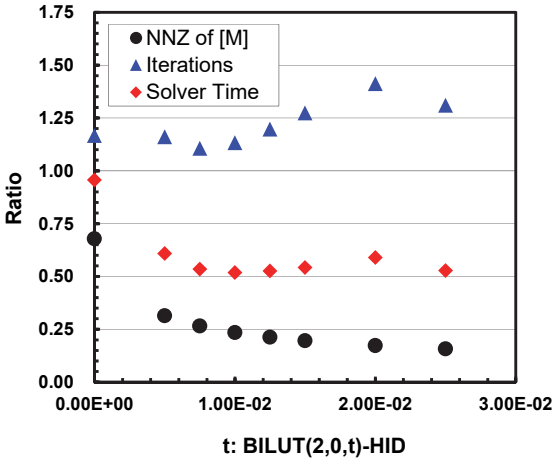


Fig.11 Effect of *t* on convergence and performance of BILUT(2,0,*t*)-HID-GPBi-CG with 240 nodes (3,840 cores) of the of the Oakleaf-FX, $E_{max}=10^{-6}$, $E_{max}=10^{+6}$, Normalized by results of BILUT(2,0,0)-LBJ-2

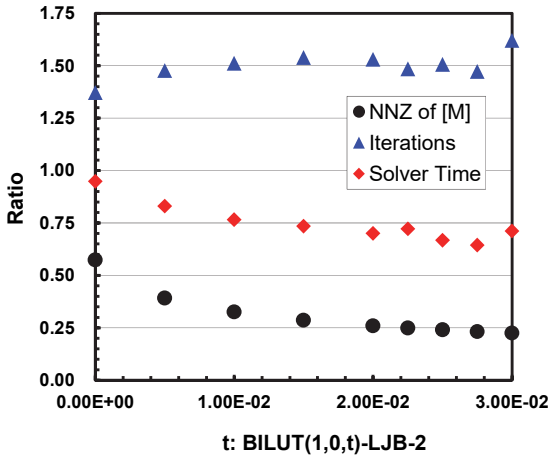


Fig.12 Effect of t on convergence and performance of BILUT(1,0,t)-LBJ-2-GPBi-CG with 240 nodes (3,840 cores) of the of the Oakleaf-FX, E_{max}=10^{-6}, E_{max}=10^{+6}, Normalized by results of BILUT(2,0,0)-LBJ-2

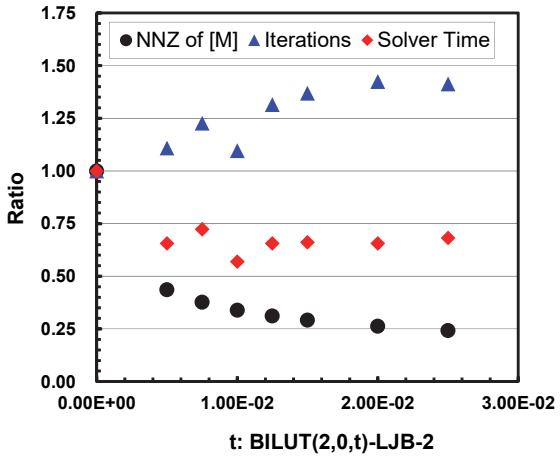


Fig.13 Effect of t on convergence and performance of BILUT(2,0,t)-LBJ-2-GPBi-CG with 240 nodes (3,840 cores) of the of the Oakleaf-FX, E_{max}=10^{-6}, E_{max}=10^{+6}, Normalized by results of BILUT(2,0,0)-LBJ-2

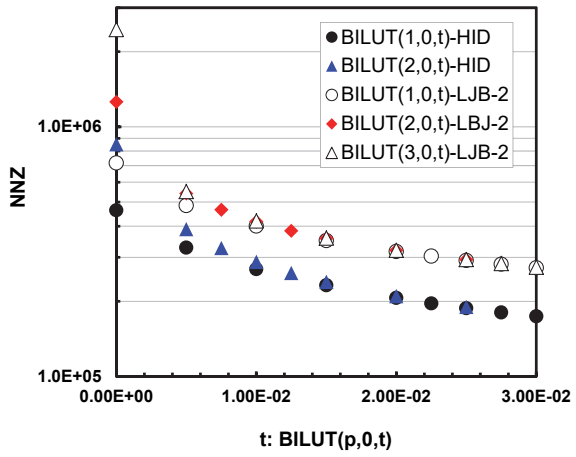
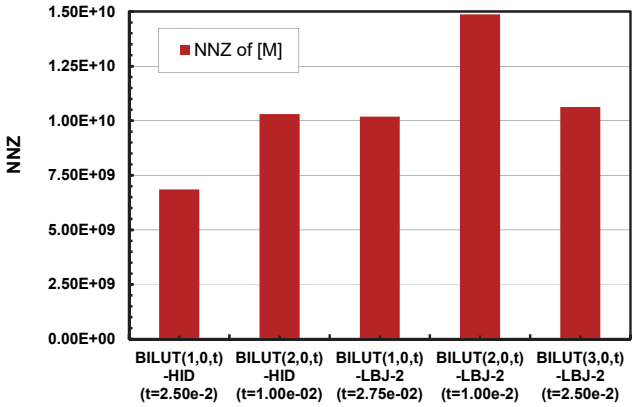
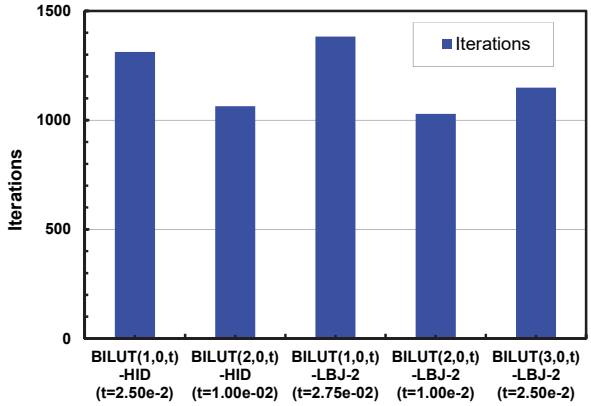


Fig.14 Relationship between t and total number of non-zero components (NNZ) in the factorized matrices for BILUT(p,d,t) ([M]) with 240 nodes (3,840 cores) of the of the Oakleaf-FX, E_{max}=10^{-6}, E_{max}=10^{+6}, d=0 for all cases

(a) NNZ of [M]



(b) Number of iterations for convergence



(c) Elapsed computation time (set-up+solver)

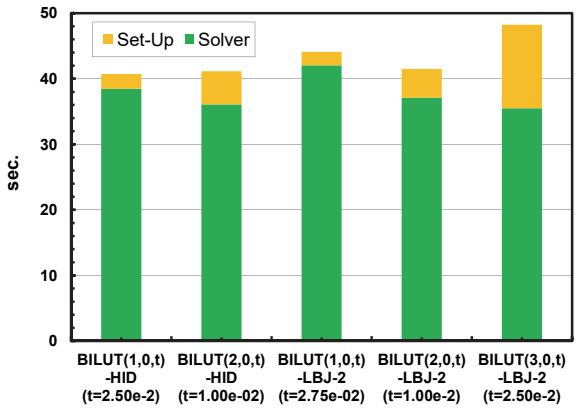


Fig.15 Results of BILUT(p,d,t)-(HID/LBJ-2)-GPBi-CG with optimum t at 240 nodes (3,840 cores) of the Oakleaf-FX, E_{max}=10^{-6}, E_{max}=10^{+6}, (a) NNZ of [M], (b) Number of iterations for convergence, (c) Elapsed computation time (set-up+solver)

Each of Fig.9 and Fig.10 shows effect of t on convergence and performance of BILUT(p,0,t)-HID, where results are normalized by the results of BILUT(2,0,0)-LBJ-2. Optimum value of t according to the time for linear solver is 2.50×10^{-2} for BILUT(1,0,t)-HID, and 1.00×10^{-2} for BILUT(2,0,t)-HID, respectively.

Each of Fig.11 and Fig.12 shows effect of t on convergence and performance of BILUT(p,0,t)-LBJ-2, where results are normalized by the results of BILUT(2,0,0)-LBJ-2. Optimum value of t according to the time for linear solver is 2.75×10^{-2} for BILUT(1,0,t)-LBJ-2, 1.00×10^{-2} for BILUT(2,0,t) -LBJ-2, and 2.50×10^{-2} for BILUT(3,0,t) -LBJ-2, respectively. Optimum values of t for BILUT(p,0,t)-HID and BILUT(p,0,t)-LBJ-2 are similar if p is equal to1 or 2.

Figure 13 shows relationship between t and total number of non-zero components (NNZ) of factorized matrices for BILUT(p,d,t) ([M]). NNZ decreases as t increases. Effect on NNZ is more significant for HID than LBJ.

Each of Fig.14(a), Fig.14(b) and Fig.14(c) compares results of BILUT(p,0,t)-HID and BILUT(p,0,t)-LBJ-2 for optimum values of t.

If we compare the results in Fig.14(c) with optimum values of t, BILUT(1,0,t)-HID, BLIUT(2,0,t)-HID, and BILUT(2,0,t)-LBJ-2 are almost competitive. Because all results in Fig.9-12 are normalized by the result of BILUT(2,0,0)-LBJ-2, we can directly compare results in these figures. If we compare ratio of computation time for linear solvers in these figures, we can get the following remarks:

- Value of ratio is generally 0.60-0.65 in BILUT(1,0,t)-HID, while that is 0.50-0.55 in BLIUT(2,0,t)-HID.
- The ratio is generally 0.75 or more in BILUT(1,0,t)-LBJ-2.
- The ratio is generally 0.65-0.70 in BILUT(2,0,t)-LBJ-2, although the optimum case of BILUT(2,0,1.0×10^{-2})-LBJ-2 hits 0.57, which is competitive with BLIUT(2,0,t)-HID as shown in Fig.14(c).

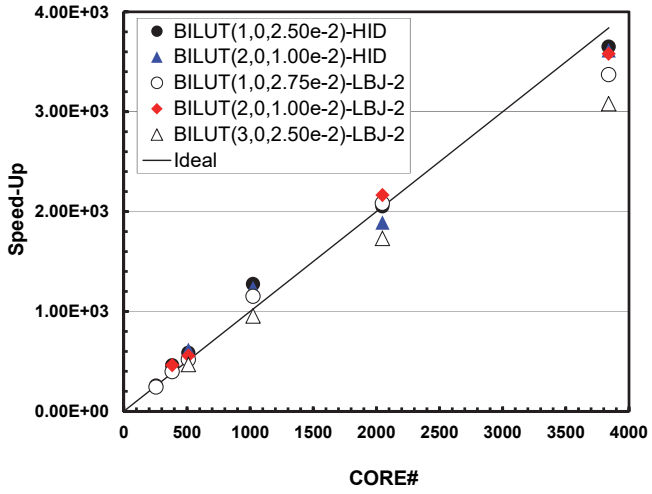
From the viewpoint of the computation time for linear solver, BILUT(2,0,t)-HID is the most robust for a wide range of values of t, although set-up time is more expensive than BILUT(1,0,t)-HID.

B. Strong Scaling

Finally, strong scaling has been applied on a 3D linear elastic problem with 400×320×240 vertices (9.216×10^7 DOF) using up to 240 nodes (3,840 cores) of the Oakleaf-FX system. Each of Fig.15(a) and Fig.15(b) shows the speed-up of elapsed time of linear solvers and setting-up by strong scaling from 256 cores to 3,840 cores. Computation time for each case is normalized by that of BILUT(1,0, 2.50×10^{-2})-HID with 256 cores. Generally speaking, BILUT(2,d,t)-HID and BILUT(2,d,t)-LBJ-2 are competitive, but BILUT(2,d,t)-HID is slightly more efficient and more robust. BILUT(2,d,t)-LBJ-2 did not converge in the case with 1,024 cores.

Between 384 cores ~ 2,048 cores, parallel performance is larger than 100%, as shown in Fig.15 (b), but less than 100% if the number of cores is more than 2,048.

(a) Speed-up



(b) Parallel Performance

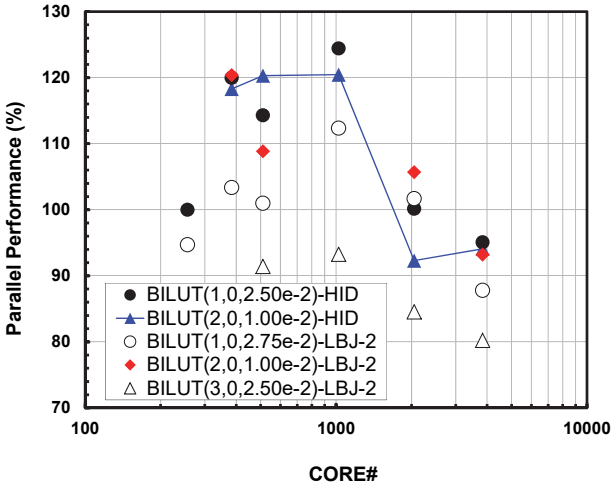


Fig.16 Strong Scalability of BILUT(p,d,t)-LBJ/HID-GPBi-CG with optimum t on the Oakleaf-FX, according to elapsed computation time (set-up+solver) for BILUT(1,0,2.5×10^{-2})-HID with 256 cores, $E_{max}=10^{-6}$, $E_{max}=10^{+6}$, d=0 for all cases

VI. Related Work

There are lot of works on ILUT preconditioning such as [2,16,17,18]. These are using a single CPU, and effects of parameters, such as dropping threshold (t in the present work) and number of non-zero off-diagonal components of factorized matrix, are evaluated. [2] is the first paper on ILUT by Saad, and efficiency and robustness are evaluated according to dropping threshold and number of non-zero off-diagonal components of the factorized matrix.

In each of [16,17,18], authors proposed method for selection of optimum combination of parameters. Mayer proposed *weighting dropping strategy* [16], which minimizes $\|A-M\|$, where M is the product of L_p and U_p, and each of L_p and U_p are lower/upper triangular matrix of ILUT factorization of A. This method is more theoretical than others, and it provides robust convergence, even if the dropping threshold is order of 10^{-1}. Gupta and George estimated the optimum parameters based on the results of other cases with

similar configuration [17]. Zhang et al. proposed *flexible factorization* [18], which controls number of non-zero off-diagonal components in factorized matrix using a simple experimental equation. This method improves distribution of eigenvalues of the factorized matrices. There are no studies related to ILUT for parallel computing, because convergence is strongly affected by method of domain decomposition, as shown in the present work.

In [3,12,19,20], the author of the present work evaluated effects of preconditioning, and domain decompositions for a wide range of problems using distributed parallel computers. Moreover, *selective blocking preconditioning* [19], *selective-fill-in/overlapping* [12,20] and *extended HID* [3] are proposed and evaluated.

VII. SUMMARY

Performance and robustness of preconditioned parallel iterative solvers based on domain decomposition for ill-conditioned problems are evaluated in the present work. The preconditioning method is based on BILUT(p,d,t) originally proposed by the author, and two types of domain decomposition procedures, LBJ (Localized Block Jacobi) and HID (Hierarchical Interface Decomposition), are considered. Developed methods are applied to Hetero3D code, which is a parallel finite-element benchmark program for solid mechanics problems, and the code provided excellent scalability and robustness up to 240 nodes (3,840 cores) of Fujitsu PRIMEHPC FX10 (Oakleaf-FX), Information Technology Center, the University of Tokyo. Generally, HID provides better performance and robustness than LBJ for a wide range of values of parameters.

Generally, ILUT(*p,d,t*) with HID provided efficient and robust convergence. Because only a single application was evaluated in the present work, the proposed method should be applied to other various kinds of applications.

Although effect of parameters was evaluated in the present work, development of the estimation of the optimum combination of parameters is very important, and it will provide excellent efficiency in scientific computing. Currently, there are no practical works in this area for parallel computing, although there are some existing works for serial computing, such as [16,17,18]. As shown in the result of BILUT(2,0,1.0×10^{-2})-LBJ-2 in Fig.12, it is generally very difficult to estimate and understand the behavior in convergence of parallel preconditioners for ill-conditioned problems. One possible criteria is the condition number of the matrix, but it is generally expensive for calculating the condition number of large scale (and distributed) sparse matrices. Extension of the methods described in [16,17,18] to parallel computation is very important for future work.

Although the author developed *extended* version of HID [3], only the original HID was evaluated in the present work. Implementation of the extended HID to Hetero3D and other applications is also an important future work. Implementation of direct linear solvers as a local preconditioner on each MPI process is also considered.

Finally, *OpenMP/MPI hybrid* version of the preconditioned iterative solver is now under development, and that will be applied to multicore/manycore clusters in near future.

ACKNOWLEDGMENT

This work is supported by JST-ANR FP3C (Collaborative Project between Japan and France on Framework and Programming for Post Petascale Computing) Project, and by Core Research for Evolutional Science and Technology (CREST), the Japan Science and Technology Agency (JST), Japan.

REFERENCES

[1] Saad, Y., Iterative Methods for Sparse Linear Systems (2nd Edition), SIAM, 2003.
[2] Saad, Y., ILUT: A dual threshold incomplete LU factorization, Numerical Linear Algebra with Applications, 387-402, 1994.
[3] Nakajima, K., Parallel Multistage Preconditioners by Extended Hierarchical Interface Decomposition for Ill-Conditioned Problems, Advances in Parallel ComputingVol.19, *From Multicores and GPU's to Petascale*, IOS press, 99-106, 2010.
[4] Nakajima, K., Parallel Preconditioning Methods for Iterative Solvers Based on BILUT(*p,d,t*), HPC in Asia Poster Session, International Supercomputing Conference (ISC'14), Leipzig, Germany, 2014.
[5] MUMPS (a MUltifrontal Massively Parallel sparse direct Solver): http://mumps.enseeiht.fr/
[6] SuperLU: http://crd-legacy.lbl.gov/~xiaoye/SuperLU/
[7] JST-ANR FP3C: Collaborative Project between Japan and France on Framework and Programming for Post Petascale Computing: http://www.systematic-paris-region.org/en/projets/fp3c
[8] Information Technology Center, The University of Tokyo: http://www.cc.u-tokyo.ac.jp/
[9] Deutsch, C.V. and Journel, A.G., GSLIB Geostatistical Software Library and User's Guide, Second Edition, Oxford Univ. Press, 1998.
[10] GeoFEM: http://geofem.tokyo.rist.or.jp/
[11] Zhang, S.L., GPBi-CG: Generalized Product-type methods based on Bi-CG for solving nonsymmetric linear systems, SIAM Journal of Scientific Computing, 18, 537-551, 1997.
[12] Nakajima, K., Strategies for Preconditioning Methods of Parallel Iterative Solvers in Finite-Element Applications on Geophysics, Advances in Geocomputing, Lecture Notes in Earth Science 119, 65-118, 2009.
[13] Henon, P. and Saad, Y., A Parallel Multistage ILU Factorization based on a Hierarchical Graph Decomposition, *SIAM Journal for Scientific Computing* 28, 2266-2293, 2007.
[14] Fujitsu: http://www.fujitsu.com/
[15] STREAM (Sustainable Memory Bandwidth in High Performance Computers): http://www.cs.virginia.edu/stream/
[16] Mayer, J., Alternative Weighted Dropping Strategies for ILUTP," SIAM Journal on Scientific Computing 27-4, 1424-1437, 2006
[17] Gupta, A. and George, T., Adaptive Techniques for Improving the Performance of Incomplete Factorization Preconditioning, SIAM Journal on Scientific Computing 32-1, 84-100, 2010.
[18] Zhang, Y., Huang. T., Jing, Y., and Li, L., Flexible incomplete Cholesky factorization with multi- parameters to control the number of nonzero elements in preconditioners, Numerical Linear Algebra with Applications 19-3, 555-569, 2012.
[19] Nakajima, K. and Okuda, H., Parallel Iterative Solvers for Simulations of Fault Zone Contact using Selective Blocking Reordering, Numerical Linear Algebra with Applications 11, 831-852, 2004.
[20] Nakajima, K., Parallel Preconditioning Methods with Selective Fill-Ins and Selective Overlapping for Ill-Conditioned Problems in Finite-Element Methods, Lecture Notes in Computer Science 4489, 1085-1092, 2007.