Strategies for Preconditioning Methods of Parallel Iterative Solvers for Finite-Element Applications in Geophysics

Kengo Nakajima

Information Technology Center, The University of Tokyo,
2-11-16 Yayoi, Bunkyo-ku, Tokyo 113-8658, Japan.

1 Background

1.1 Why Preconditioned Iterative Solvers?

Solving large-scale systems of linear equations $[A]\{x\}=\{b\}$ is one of the most expensive and critical processes in scientific computing. In particular, for simulation codes based on the finite-element method (FEM), most of the computational time is devoted to solving linear equation systems with sparse coefficient matrices. For this reason, a significant proportion of scalable algorithm research and development is aimed at solving these large, sparse linear systems of equations on parallel computers. Sparse linear solvers can be broadly classified as being either *direct* or *iterative*.

Direct solvers, such as Gaussian elimination and LU factorization, are based on a factorization of the associated sparse matrix. They are extremely robust and yield the exact solution of $[A]\{x\}=\{b\}$ after a finite number of steps without round-off errors. However, their memory requirements grow as a nonlinear function of the matrix size because initially zero components of the original matrix fill in during factorization. In contrast, iterative methods are memory scalable. Iterative methods are therefore the only choice for large-scale simulations by massively parallel computers. While iterative methods are memory scalable, a disadvantage is that their convergence can be slow or they can fail to converge. The rate of convergence of iterative methods depends strongly on the spectrum of the coefficient matrix. Hence, iterative methods usually involve a second matrix that

H. Xing, *Advances in Geocomputing*, Lecture Notes in Earth Sciences 119,
DOI 10.1007/978-3-540-85879-9_3

transforms the coefficient matrix to a matrix with a more favorable spectrum. This transformation matrix is called a *preconditioner*. The improvement in the convergence of an iterative method yielded by the application of an effective preconditioner outweighs the extra cost of constructing and applying it. Indeed, without a preconditioner the iterative method may even fail to converge.

In preconditioned iterative methods, the original linear equation:

$$[A]\{x\}=\{b\} \tag{1}$$

is transformed into the following Eq. (2) using the preconditioner (or preconditioning matrix) $[M]$:

$$[\tilde{A}]\{x\}=\{\tilde{b}\},\quad [\tilde{A}]=[M]^{-1}[A],\quad \{\tilde{b}\}=[M]^{-1}\{b\} \tag{2}$$

Equation (2) has the same solution as Eq. (1), but the spectral properties of the coefficient matrix $[\tilde{A}]=[M]^{-1}[A]$ may be more favorable, facilitating a faster convergence.

Various types of preconditioners have been proposed, developed and applied. The simplest method is called *diagonal scaling* or the *point Jacobi* method, where $[M]$ is given by the diagonal components of the original coefficient matrix $[A]$. Jacobi, Gauss-Seidel and SOR type stationary iterative methods are also well-known preconditioners, while preconditioners using various types of polynomials have also been widely used (Barrett et al. 1994).

Au: "Barret" has been changed to "Barrett" in all occurrences, in order to match with the reference. Is this OK?

The incomplete lower-upper (*ILU*) and incomplete Cholesky (*IC*) factorization methods are the most popular preconditioning techniques for accelerating the convergence of Krylov iterative methods (Barrett et al. 1994). These ILU/IC methods are based on *LU/Cholesky factorization* used in direct solution techniques. LU factorization is applicable to general un-symmetric matrices, while Cholesky is applicable to symmetric matrices. In *LU/Cholesky factorization*, many *fill-ins* are introduced during the factorization process, and so the factorized matrix can be dense even if the original matrix is sparse. **ILU(p)/IC(p)** is an incomplete factorization in which *p*-th-order fill-ins are allowed. Larger values of *p* provide a more accurate factorization and usually lead to robust preconditioning, but are more expensive in both memory and CPU time. In many engineering applications, **ILU(0)/IC(0)** is widely used where there are no fill-ins and the non-zero pattern of the original coefficient matrix is maintained in the factorized matrix.

1.2 *Selective Blocking* Preconditioning for Contact Problems

1.2.1 GeoFEM Project

From 1999 to 2002, the author developed parallel iterative solvers and preconditioning methods for geophysics problems in the *GeoFEM* project,[1] which develops a parallel finite-element platform for solid earth simulation on the *Earth Simulator*.[2] One of the most important applications of GeoFEM is the simulation of the stress accumulation process at plate boundaries (faults), which is critical for estimating the earthquake generation cycle (Figs. 1 and 2). A fine resolution (less than 1 km) is required around zones with higher stress accumulations, and so more than hundreds of millions of meshes may be required for detailed simulations. In this type of simulation, material, geometric and boundary nonlinearity should be considered. Among these, boundary nonlinearity due to the contact of faults is the most critical. In GeoFEM, the augmented Lagrange method (ALM) and penalty method are implemented, with a large penalty number, λ, introduced for constraint conditions around faults (Iizuka et al. 2000). The nonlinear process is solved iteratively by the Newton-Raphson (NR) method. A large λ ($\sim 10^3 \times$ Young's modulus) can provide an accurate solution and fast nonlinear convergence for NR processes, but the condition number of the coefficient matrices of the corresponding linear equations is large, and several iterations are required for the convergence of iterative solvers (Fig. 3). Therefore, a robust preconditioning method is essential for such ill-conditioned problems.

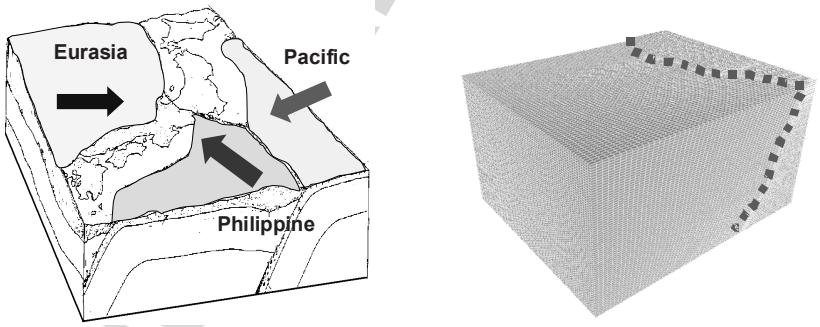


Fig. 1 Subductive plate boundaries (faults) around Japanese Islands and an example of the finite-element model

[1] http://geofem.tokyo.rist.or.jp/
[2] http://www.es.jamstec.go.jp/

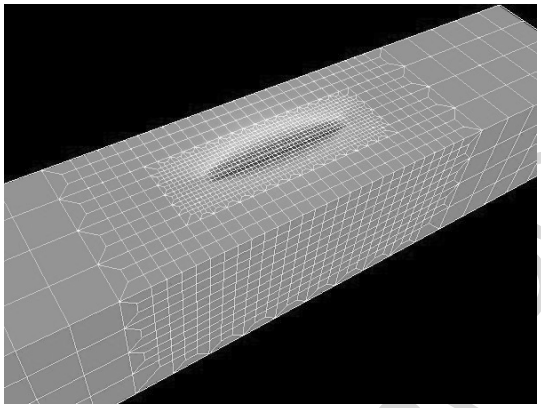


Fig. 2 Example of the finite-element model with locally refined meshes for a transcurrent fault (movie available on accompanying DVD)

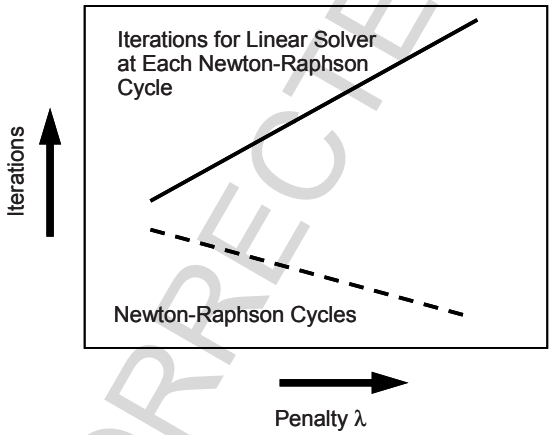


Fig. 3 Typical relationship between λ (penalty number) and the required number of iterations in contact simulations by ALM (Iizuka et al. 2000)

1.2.2 Selective Blocking

Selective blocking is a special preconditioning method developed for this type of application by the author on GeoFEM's framework (Nakajima 2003, Nakajima and Okuda 2004). In this method, finite element nodes in the same contact group coupled through penalty constraints are placed into a large block (selective block or super node) (Fig. 4). For symmetric positive definite matrices, preconditioning with *block* incomplete Cholesky factorization *using selective blocking* (SB-BIC) yields an excellent performance and robustness (Nakajima 2003, Nakajima and Okuda 2004). Details of the parallel iterative solvers of GeoFEM and algorithm of selective blocking are described in Appendices 1 and 2 of this chapter.

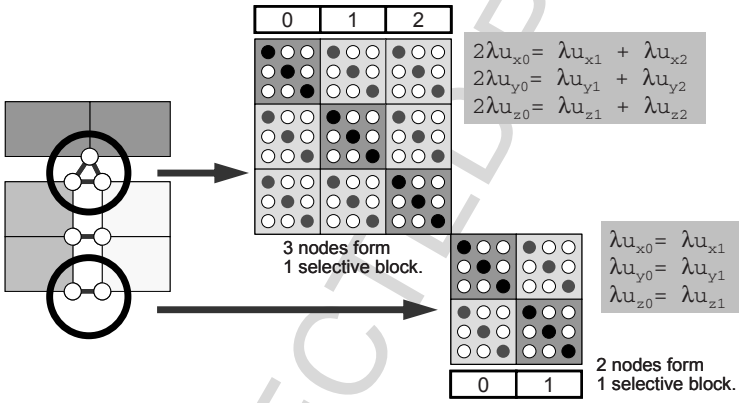


Fig. 4 Matrix operation of nodes in contact groups for *selective blocking* preconditioning

1.3 Overview of this Work

Contact phenomena are one of the most important and critical issues in various types of scientific and engineering problems. In previous works (Nakajima 2003, Nakajima and Okuda 2004), the numbers of nodes in contact groups are consistent, and conditions for infinitesimal deformation have been also assumed, as shown in Fig. 5a. With this approach, the positions of nodes do not change and a consistent relationship among nodes in contact groups is maintained during the simulation. Moreover, a special partitioning method, where all nodes in the same contact group are located in the same domain, has been applied, as shown in Appendix 2. However, this approach is not therefore flexible, and cannot be applied to fault contact simulations with large slip/deformation and to simulations of assembly structures in engineering fields (Fig. 6), where the numbers and positions of nodes in contact groups may be inconsistent, as shown in Fig. 5b. In this situation, number of finite-element nodes in each *selective block* might be very large. If the size of selective block is more than 10^3, preconditioning with full LU factorization for each block is very expensive.

In (Nakajima 2007a), new parallel preconditioning methods for this type of general contact problem have been developed. These methods comprise two parts: One part is a preconditioning method with *selective fill-ins*, in which *fill-ins* of higher order are introduced only for nodes connected to special contact-condition elements.

The other part is the extension of overlapped elements between domains. It is widely known that convergence of parallel finite-element applications with preconditioned iterative solvers strongly depends on method of domain decomposition. In (Nakajima 2007a), the *selective overlapping* method was proposed, which extends the layers of overlapped elements according to the information of the special elements used for contact conditions. Both methods are based on the idea of *selective blocking*, but are more general and flexible than that approach.

These methods are very unique, because dropping rules of the preconditioning matrices are defined according to properties of individual finite-element and features of finite-element applications before assembling entire coefficient matrices.

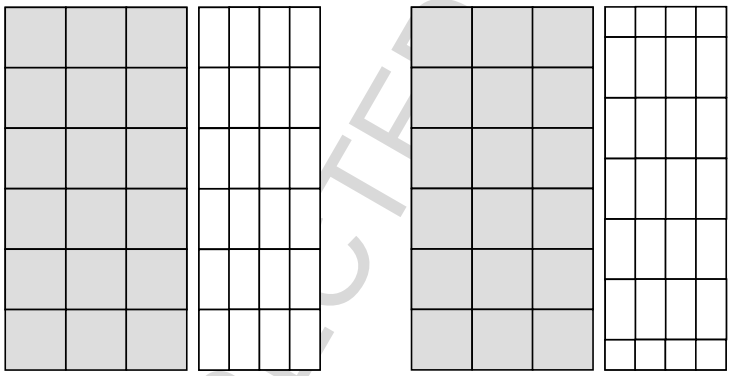
In addition, the following two methods are further introduced in this work:

- Local reordering in distributed data
- Hierarchical Interface Decomposition (HID) (Henon and Saad 2007)

HID provides a method of domain decomposition with robustness and scalability for parallel ILU/IC preconditioners. The robustness and efficiency of HID and *selective overlapping* are compared in this work.

In the following part of this chapter, these four methods (selective fill-ins, selective overlapping, local reordering and HID) are reviewed in detail. These methods are implemented as preconditioners of iterative solvers for parallel finite-element applications for ill-conditioned problems. Parallel codes are based on the framework for parallel FEM procedures of GeoFEM, and the GeoFEM's local data structure (ref. Appendix 1) is applied.

The results of example problems with contact conditions using 64-core PC clusters are shown. Finally, the developed methods are applied to general ill-conditioned problems for problems with heterogeneous material properties.



(a) Consistent (b) Inconsistent

Fig. 5 Consistent and inconsistent node numbers at contact surfaces in FEM models applied to contact simulations

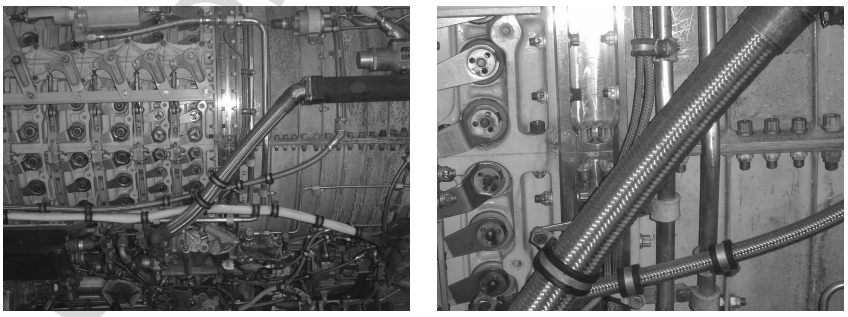


Fig. 6 Example of an assembly structure: Jet Engine

2 Various Approaches for Parallel Preconditioning Methods in Ill-Conditioned Problems

2.1 Selective Fill-Ins

The *selective blocking* preconditioning method (Nakajima and Okuda 2004) is a robust and efficient preconditioning method for contact problems. However, it can only be applied in a very limited number of situations, as shown in the previous section.

Incomplete LU factorization with p-th-order fill-in (**ILU(p)**) preconditioning methods are widely used for various types of applications (Saad 2003). The higher the order of fill-ins (**p**) is, the more robust the preconditioner will be, but this normally comes at the cost of being computationally more expensive. The required memory for coefficient matrices increases by a factor of from 2 to 5 if the order of fill-ins (**p**) increases from 0 to 1, or from 1 to 2 (Nakajima and Okuda 2004).

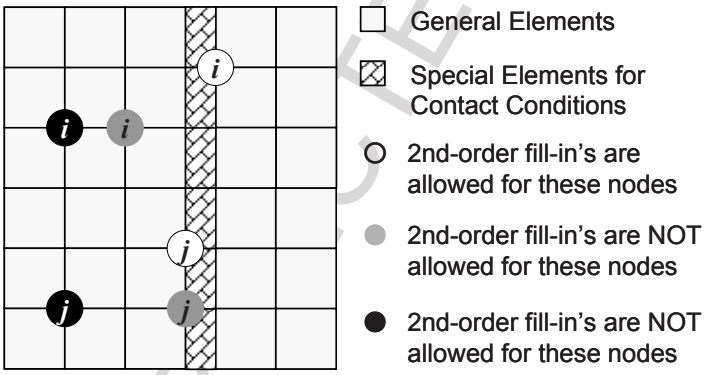


Fig. 7 Example of the **ILU(1+)** preconditioning technique

In (Nakajima 2007a), new preconditioning methods for general contact problems have been developed. The first approach is a preconditioning method with *selective fill-ins*, called **ILU(p+)**. Figure 7 describes the principle of **ILU(p+)**. Denoting the i,j-th component of the preconditioner matrix by m_{ij} in **ILU(p+)**, (p+1)-th order fill-ins are allowed for m_{ij} such that both the *i-th* and *j-th* nodes are connected to special contact-condition elements, such as *master-slave* type elements (Iizuka et al. 2000). In Fig. 7,

second-order fill-ins can be allowed for all three *i-j* pairs, according to graphical connectivity information. However, in the **ILU(p+)** preconditioning approach**,** only the white circles are allowed to generate second-order fill-ins.

This approach closely resembles that of *selective blocking*, in which full LU factorization is applied to nodes in contact groups, but is much more general and flexible. Since constraint conditions are applied to the nodes that are connected to special elements through penalty terms, *selective* ILU factorization with higher order fill-ins for these nodes is expected to provide robust convergence with efficiency. In (Washio et al. 2005), a preconditioning method with block ILU factorization is proposed for coupled equations of incompressible fluid flow and solid structure. Different orders of fill-ins are applied to velocity and pressure components to generate block ILU factorization of coefficient matrices. **ILU(p+)** is very similar to this idea.

Figure 8 describes the model used for the validation of the developed preconditioning methods. This problem simulates general contact conditions, in which the positions and number of nodes on contact surfaces are inconsistent. In this model there are four blocks of elastic material that are discretized into cubic tri-linear type finite-elements. Each block is connected through elastic truss elements generated at each node on the contact surfaces. The truss elements together take up the form of a cross, as shown in Fig. 8. In the present case, the elastic coefficient of the truss elements is set to 10^3 times that of the solid elements, which corresponds to the coefficient λ $(=10^3)$ for constraint conditions of the augmented Lagrangian method (ALM). Poisson's ratio is set to 0.25 for the cubic elements.

Symmetric boundary conditions are applied at the x=0 and y=0 surfaces, while a Dirichlet fixed condition for deformation in the direction of the z-axis is applied to z=0 surfaces. Finally, a uniform distributed load in the direction of the z-axis is applied to $z=Z_{max}$ surfaces. This problem lies in the area of linear elasticity, but the coefficient matrices are particularly ill-conditioned, and so this problem provides a good simulation of nonlinear contact problems (Nakajima 2003, Nakajima and Okuda 2004).

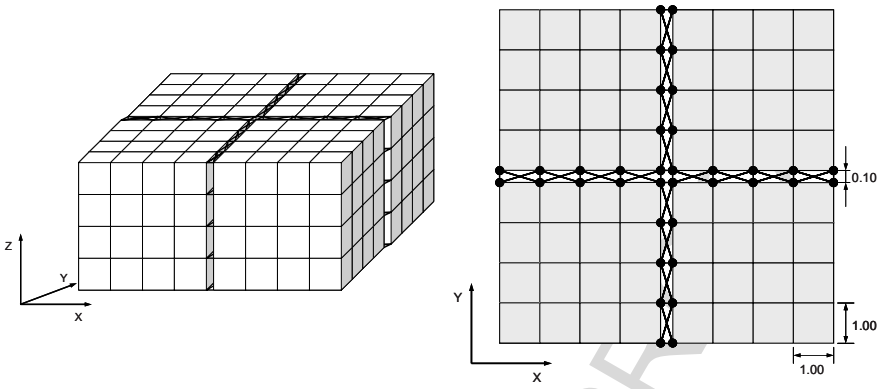


Fig. 8 Elastic blocks connected through truss elements

The plots in Fig. 9 display results describing the performance of four preconditioning techniques for the problem in linear elasticity described above. All calculations were performed using a single core of AMD Opteron 275 (2.2 GHz)[3] with PGI FORTAN90 compiler.[4] Each block in Fig. 8 has 8,192 (=16×16×32) cubes, where the total problem size has 107,811 degrees of freedom (DOF). Generalized product-type methods based on Bi-CG (GPBi-CG) (Zhang 1997) for general coefficient matrices have been applied as an iterative method, although the coefficient matrices for this problem are positive indefinite. Each node has three DOF in each axis in 3D solid mechanics; therefore, ***block*** **ILU (BILU)** type preconditioning (Nakajima 2003, Nakajima and Okuda 2004) has been applied.

BILU(1+), in which additional *selective fill-ins* to **BILU(1)** have been applied for nodes connected to special elements (elastic truss elements in Fig. 8), provides the most robust and efficient convergence. **BILU(p)** provides faster convergence the larger the value of p, as shown in Fig. 9b, but is also more computationally expensive with increasing p, as shown in Fig. 9c, where the number of off-diagonal components in preconditioning matrices [*M*] is described. **BILU(1)** and **BILU(1+)** are competitive, but **BILU(1+)** provides a better convergence rate.

[3] http://www.amd.com/

[4] http://www.pgroup.com/

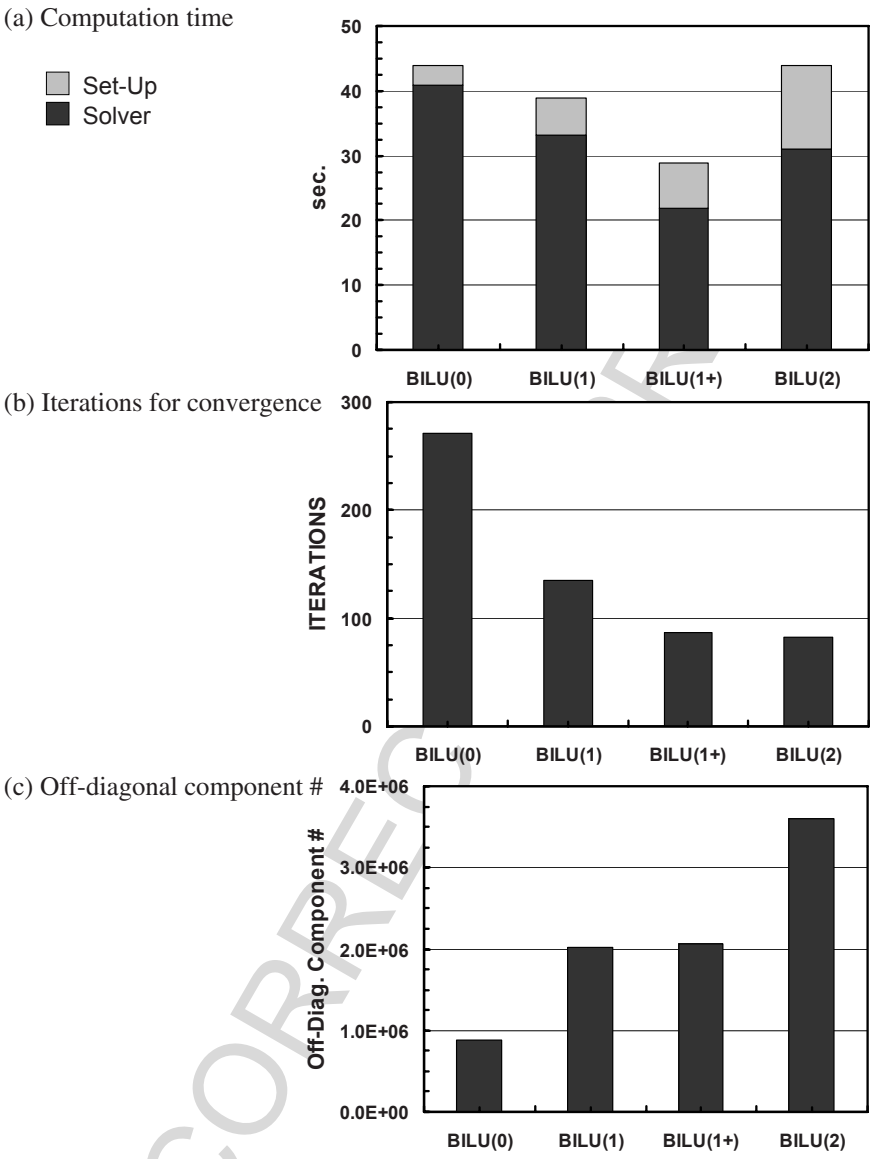


Fig. 9 Results for the problem in linear elasticity, whose configuration is given in Fig. 8, considering simple cube geometries of 107,811 DOF with contact conditions on a single core of AMD Opteron 275 (2.2 GHz) with the PGI FORTRAN90 compiler

2.2 Selective Overlapping

The second approach proposed here is the extension of overlapped zones between domains for parallel computing. The GeoFEM local data structure, which has been applied in previous works (Nakajima 2003, Nakajima and Okuda 2004), is node-based with a single layer of overlapped elements (the depth of overlapping is 1) and is appropriate for parallel iterative solvers with block Jacobi-type localized preconditioning methods. Figure 10 shows an example of the local data for contact problems, in which the depth of overlapping is 1.

In (Nakajima 2007a), a larger number of layers of overlapped elements were considered to improve the robustness of parallel preconditioners. Generally speaking, a larger depth of overlapped layers provides faster convergence in block Jacobi-type localized preconditioning methods, but at the expense of increasing computation and communication costs (Nakajima 2005).

In (Nakajima 2007a), the *selective overlapping* method was proposed. As is illustrated in Fig. 11, for the process of extending overlapped areas, this method gives priority to those nodes connected to special contact-condition elements. In particular, in *selective overlapping*, the extension of overlapping to nodes that are *not* connected to special contact-condition elements is *delayed*. For example, the *hatched* elements shown in the selective overlapping plots of Fig. 11 would be included as extended overlapped elements in a conventional overlapping extension. However, in selective overlapping, the extension of overlapping to include these elements is delayed, and is instead performed at the next stage of overlapping. Thus, the increases in computation and communication costs due to the extension of the overlapped elements are reduced.

This idea is also an extension of the idea of *selective blocking*, and is also based on the idea of special partitioning strategy for contact problems, developed in (Nakajima and Okuda 2004). The convergence rate of parallel iterative solvers with block Jacobi-type localized preconditioning is generally poor, because the *edge-cut* may occur at inter-domain boundary edges that are included in contact groups (Nakajima and Okuda 2004). All nodes in the same contact group should be in the same domain in order to avoid such edge-cuts. Because the constraint conditions are applied to those nodes that are connected to special elements through penalty terms, the *selective* extension of overlapping for these nodes is expected to provide robust convergence with efficiency.

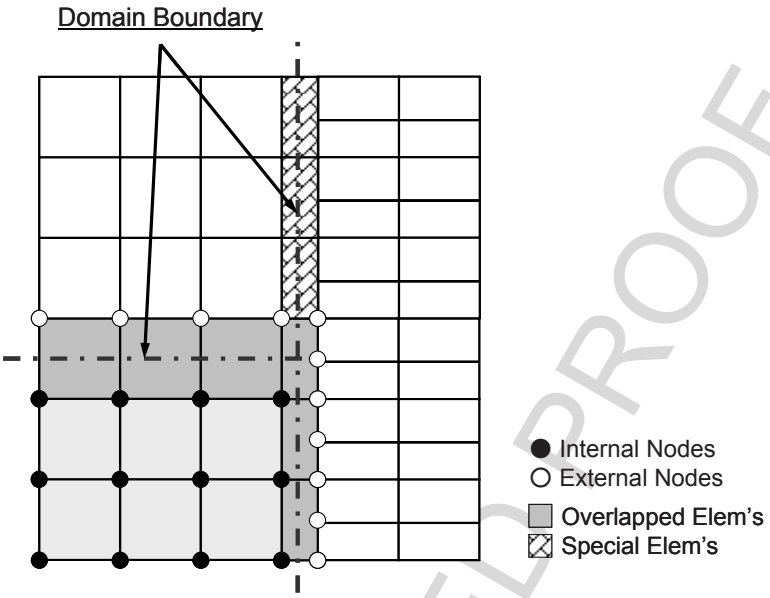


Fig. 10 Example of GeoFEM's local data structure for contact problems

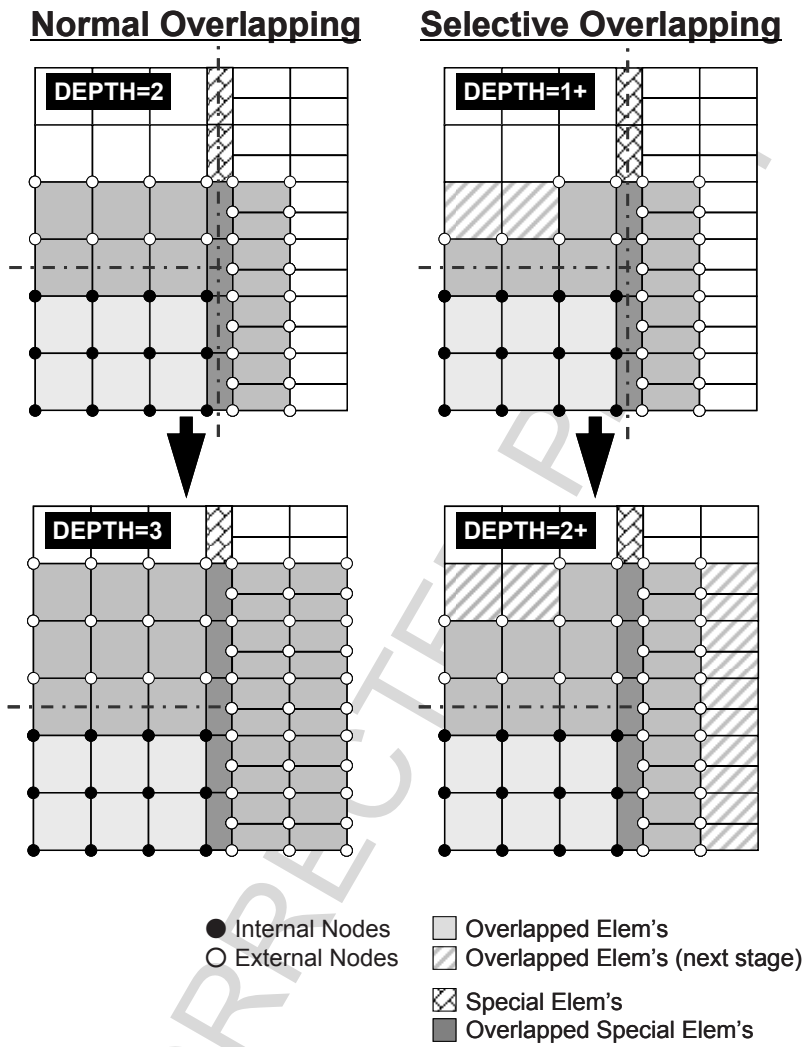


Fig. 11 Example of *selective overlapping*, precedence for extensions of overlapped layers is given to nodes connected to special contact-condition elements

2.3 Local Reordering in Distributed Data

It is widely known that the reordering of vertices strongly affects the convergence of iterative solvers with ILU-type preconditioners (Nakajima 2007b).

As shown in Appendix 1, nodes in each local mesh data of the GeoFEM data structure are classified into the following *three* categories from the viewpoint of message passing:

- 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 12 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. 12. According to this numbering method, the bandwidth of local sparse coefficient matrices including the external nodes is relatively large, as shown in Fig. 12. Coefficient matrices with larger bandwidth usually provide slower convergence for iterative solvers with ILU-type preconditioning methods (Saad 2003). As long as block Jacobi-type fully localized preconditioning methods in Nakajima (2003) are adopted, this effect of bandwidth is rather smaller. But if overlapping among domain is applied, this effect may be more significant.

In this work, a *global* numbering is introduced, where both the internal and external nodes in each domain are reordered according to their original global ID. Figure 13 shows local data meshes obtained through this *global* numbering, and corresponding local coefficient matrices. It may be noted that the bandwidth of local coefficient matrices is smaller than that of original matrices in Fig. 12.

The Reverse Cuthill-Mckee (RCM) method is a well-known reordering technique, which is also suitable for reducing the bandwidth of sparse matrices (Nakajima 2003, Saad 2003). In the previous works (Nakajima 2003, Nakajima 2007b), RCM reordering has been applied to internal nodes for parallel efficiency. In this work, the following two types of approaches have been applied:

- RCM is applied only to internal nodes (Fig. 14)
  - local numbering with *RCM-internal*
- RCM is applied to both internal and external nodes (Fig. 15)
  - local numbering with *RCM-entire*

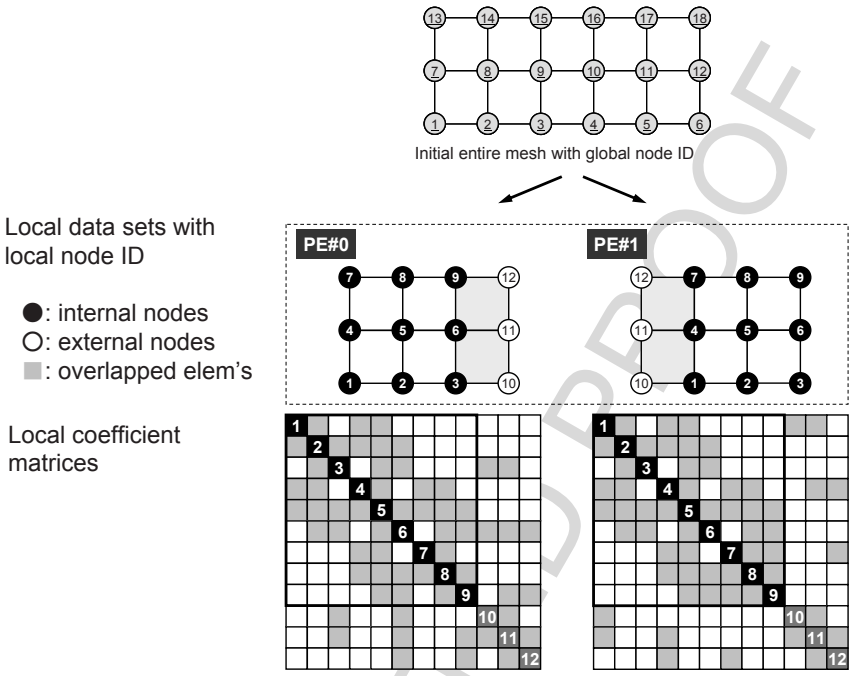


Fig. 12 An initial entire mesh with global node ID, local domains with local node ID and local coefficient matrices for two domains

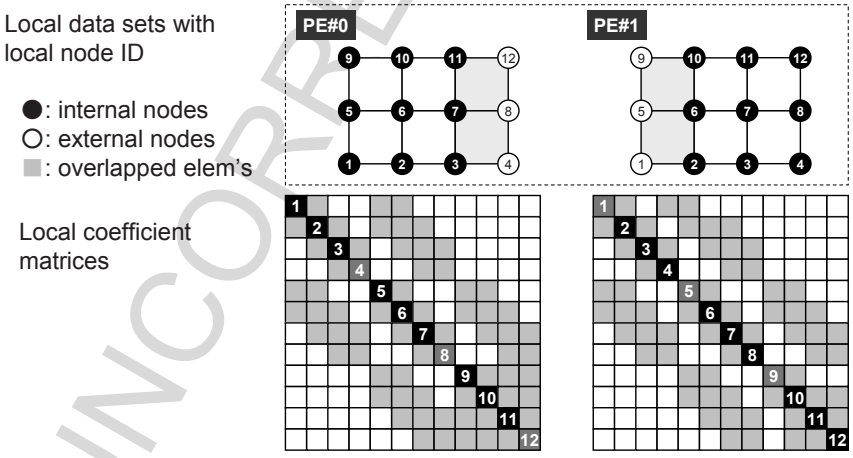


Fig. 13 Local domains and coefficient matrices according to *global* numbering

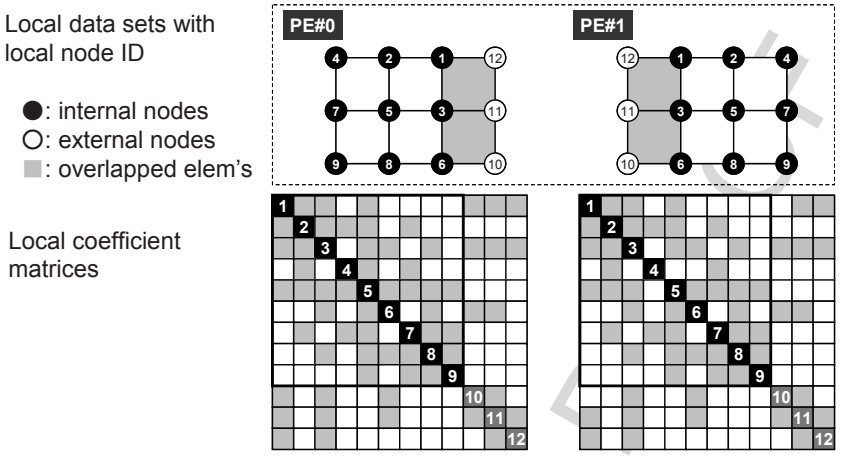


Fig. 14 Local domains and coefficient matrices with *RCM-internal* reordering

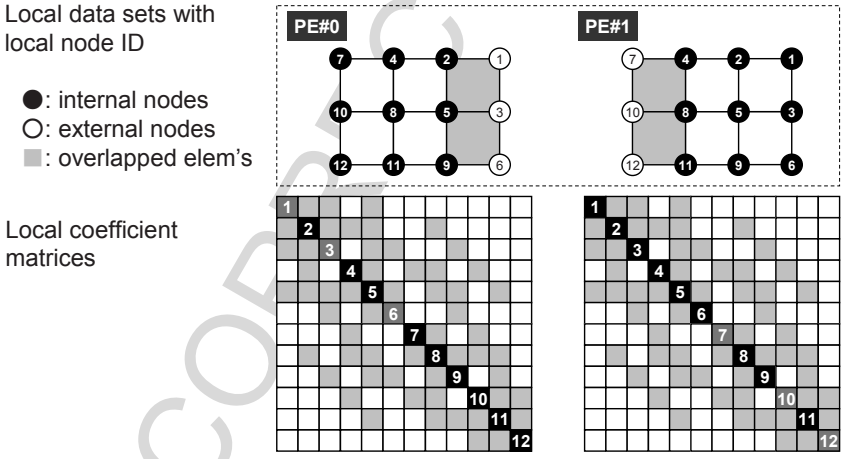


Fig. 15 Local domains and coefficient matrices with *RCM-entire* reordering

2.4 HID (Hierarchical Interface Decomposition)

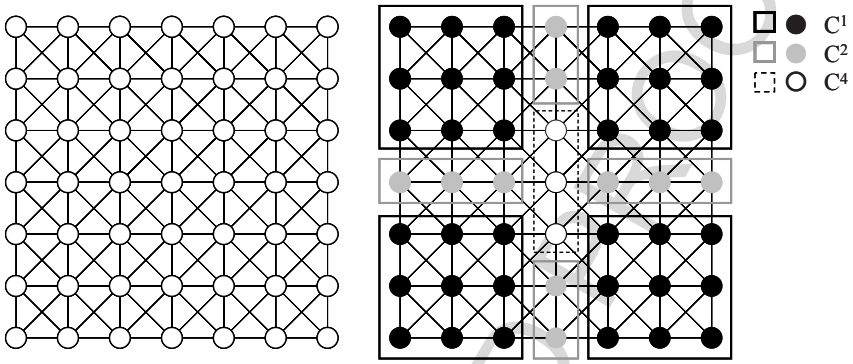
The Parallel Hierarchical Interface Decomposition Algorithm (PHIDAL) provides robustness and scalability for parallel ILU/IC preconditioners (Henon and Saad 2007). PHIDAL is based on defining "*hierarchical interface decomposition* (HID)". The HID process starts with a partitioning of the graph, with one layer of overlap. The "levels" are defined from this partitioning, with each level consisting of a set of vertex groups. Each vertex group of a given level is a *separator* for vertex groups of a *lower* level. The incomplete factorization process proceeds by "level" from lowest to highest. Due to the separation property of the vertex groups at different levels, this process can be carried out in a highly parallel manner. In (Henon and Saad (2007), the concept of *connectors* (small connected sub-graphs) of different *levels* and *keys* are introduced for the purposes of applying this idea to general graphs as follows:

- Connectors of *level-1* (C^1) are the sets of interior points. Each set of interior points is called a *sub-domain*.
- A connector of *level-k* (C^k) (k>1) is adjacent to k *sub-domains.*
- No C^k is adjacent to any other connector of level-k.
- *Key*(*u*) is the set of sub-domains (connectors of level-1, C^1) connected to vertex *u*.

Figure 16 shows the example of the partition of a 9-point grid into 4 domains. In this case, there are 4 connectors of level-1 (C^1, sub-domain), 4 connectors of level-2 (C^2) and 1 connector of level-4 (C^4). Note that different connectors of the same level are not connected directly, but are separated by connectors of higher levels. These properties induce a block structure of the coefficient matrix [*A*] through reordering the unknowns by this decomposition. 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. 17. This block structure leads to a natural parallelism if ILU/IC decompositions or forward/backward substitution processes are applied. Figure 18 provides algorithms for the construction of independent connectors (Henon and Saad 2007). Thus, HID/PHIDAL-based ILU/IC preconditioners can consider the global effect of external domains in parallel computations, and are expected to be more robust than block Jacobi-type localized ones. In this work, HID and *selective overlapping* are compared from this point of view.

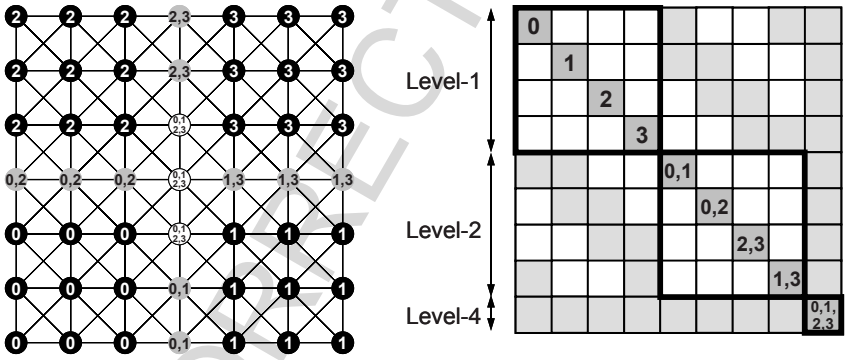
In (Nakajima 2007b), GeoFEM's original partitioner for domain decomposition was modified so that it could create a distributed hierarchical data structure for HID. Each *sub-domain* (interior vertices, connectors of

level-1) is assigned to an individual *domain*, which corresponds to each MPI process. Higher-level connectors are distributed to each domain so that load-balancing can be attained and communications can be minimized. Figure 19 shows an example of the final partition of a 9-point grid into 4 sub-domains.



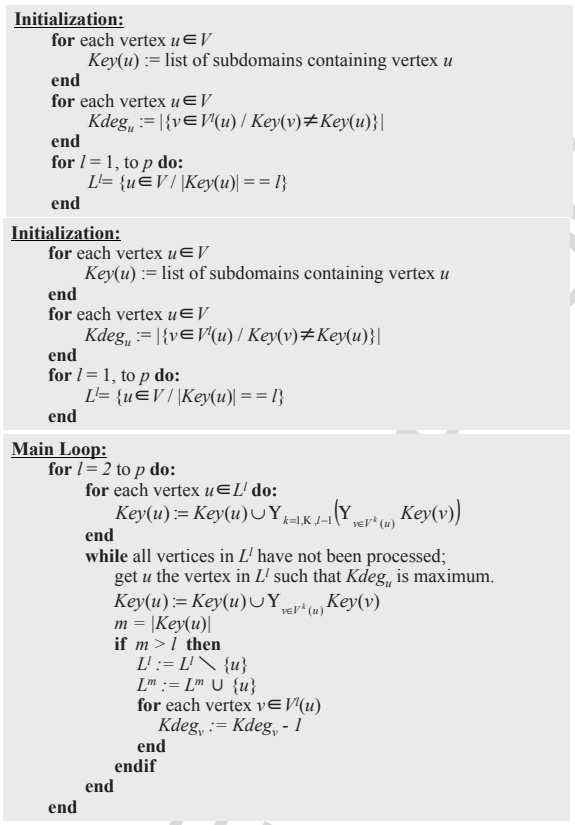
(a) Initial entire grid (b) *Connectors* and *levels*

Fig. 16 HID partitioning of a 9-point grid into 4 sub-domains



(a) Domain decomposition (*connectors* and *keys*) (b) matrix and non-empty blocks

Fig. 17 Domain/block decomposition of the coefficient matrix according to HID reordering

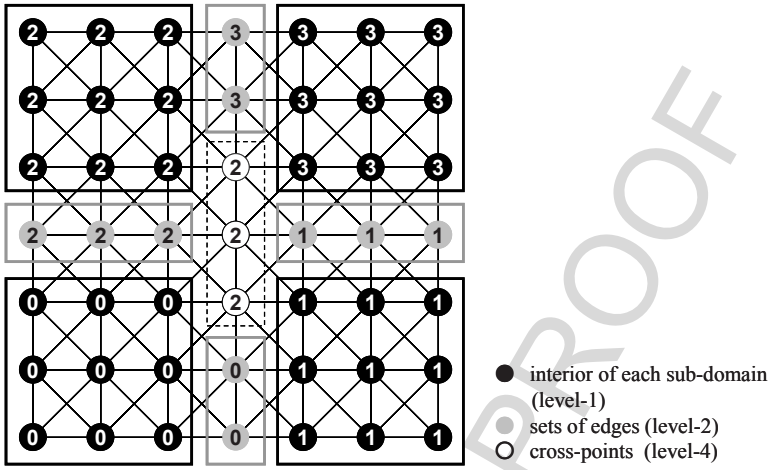


Initialization:
for each vertex $u \in V$
$Key(u)$:= list of subdomains containing vertex u
end
for each vertex $u \in V$
$Kdeg_u := |\{v \in V^n(u) \,/\, Key(v) \neq Key(u)\}|$
end
for $l = 1$, to p **do:**
$L^l = \{u \in V \,/\, |Key(u)| = = l\}$
end

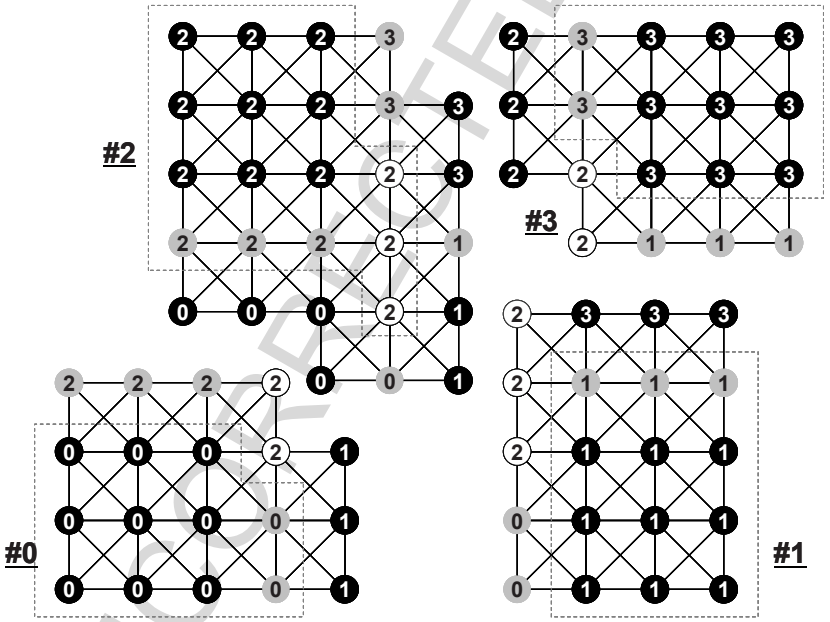
Initialization:
for each vertex $u \in V$
$Key(u)$:= list of subdomains containing vertex u
end
for each vertex $u \in V$
$Kdeg_u := |\{v \in V^n(u) \,/\, Key(v) \neq Key(u)\}|$
end
for $l = 1$, to p **do:**
$L^l = \{u \in V \,/\, |Key(u)| = = l\}$
end

Main Loop:
for $l = 2$ to p **do:**
for each vertex $u \in L^l$ **do:**
$Key(u) := Key(u) \cup \Upsilon_{k=1,K,l-1}\left(\Upsilon_{v \in V^k(u)} Key(v)\right)$
end
while all vertices in L^l have not been processed;
get u the vertex in L^l such that $Kdeg_u$ is maximum.
$Key(u) := Key(u) \cup \Upsilon_{v \in V^k(u)} Key(v)$
$m = |Key(u)|$
if $m > l$ **then**
$L^l := L^l \setminus \{u\}$
$L^m := L^m \cup \{u\}$
for each vertex $v \in V^n(u)$
$Kdeg_v := Kdeg_v - 1$
end
endif
end
end

Fig. 18 Algorithms for HID processes (Henon and Saad 2007)



(a) Final partition (number's correspond to ID of partition (0-3))



(b) Distributed local data sets with external vertices

Fig. 19 The final partition of a 9-point grid into 4 domains

3 Examples: Contact Problems

3.1 Effect of Selective Fill-Ins and Selective Overlapping

The plots in Fig. 20 compare the effect of different overlapping strategies on results obtained for the problem in linear elasticity described by Fig. 8. Results were obtained using 64 cores of an AMD Opteron 275 cluster with a PGI FORTAN90 compiler and Pathscale MPI[5] connected through an Infiniband[6] network. Each block in Fig. 8 has 250,000 (=50×50×100) cubes, yielding a total problem size of 3,090,903 DOF. The effect of the extension of overlapping is evaluated for **BILU(1)**, **BILU(1+)**, and **BILU(2)**. Here **BILU(p)-(d)** means **BILU(p)** preconditioning, where the depth of overlapping is equal to **d**. The local data with a single layer of overlapped elements, shown in Fig. 10, is applied to both of **(d=0)** and **(d=1)**. In **(d=0)**, effect of *external nodes* are *not* considered at all during ILU/IC decompositions and forward/backward substitution processes. Therefore, **BILU(p)-(0)** corresponds to *pure* block Jacobi-type localized preconditioning method, which provides excellent parallel efficiency, but is not robust for ill-conditioned problems. In **(d=1)**, effect of external nodes are considered in preconditioning and forward/backward substitution processes.

Partitioning was applied in an RCB (recursive coordinate bisection) manner (Simon 1991), and the entire domain has been partitioned into 64 local data sets. The initial local numbering procedure shown in Fig. 12 has been applied to each local data set.

Generally speaking, the convergence rate is improved by the extension of overlapping (Fig. 20b). This is particularly significant when the depth of overlapping (**d**) is increased from **(d=0)** and **(d=1)** to **(d=1+),** because *edge-cuts* may occur at truss elements for contact conditions if the depth of overlapping is 0 or 1. However, the decrease in the number of iterations required for convergence is comparatively small for further rises in **d** if the depth of overlapping is greater than 2.

It can also be seen that the number of off-diagonal components of the preconditioned matrices [*M*] increases as the depth of overlapping increases (Fig. 20c). Finally, computations using large depths of overlapping are more expensive, as may be seen in Fig. 20a, where the computational time increases as the depth of overlapping increases from values larger than 2 (Fig. 20a). Methods **BILU(1)-(1+)** and **BILU(1+)-(1+)** offer the best performance, and these two methods are closely matched.

[5] http://www.pathscale.com/

[6] http://www.infinibandta.org/

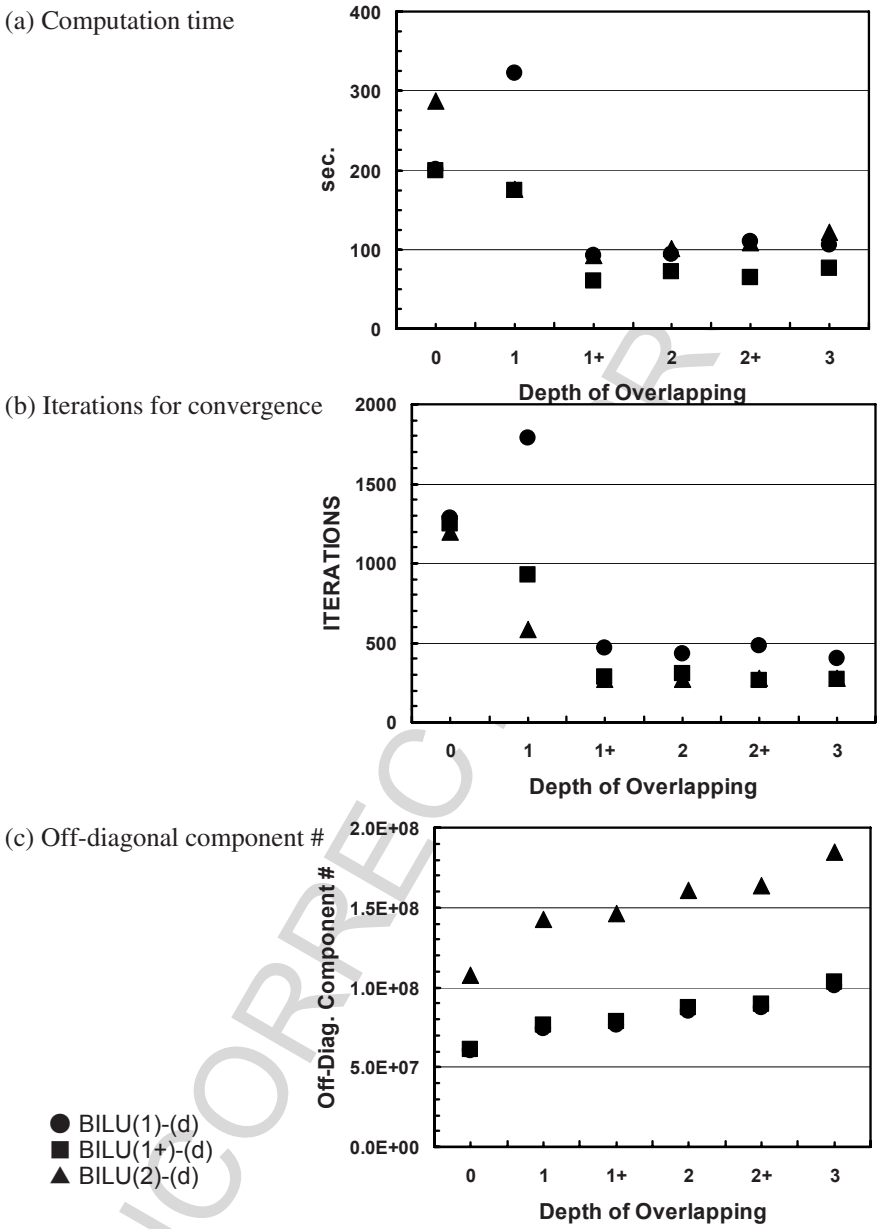


Fig. 20 Results detailing the effect of overlapping for the model problem considering simple cube geometries of 3,090,903 DOF with contact conditions in linear elasticity described by Fig. 8 on 64 cores of AMD Opteron 275 cluster using the PGI compiler

3.2 Effect of Local Reordering

The plots in Fig. 21 show the effect of the extension of overlapping for **BILU(1+)**, which was the best performing method of Sect. 3.1, on various types of orderings/numberings for the same test problem considered in Sect. 3.1, The numbering strategies considered here are the initial local numbering (Fig. 12), global numbering (Fig. 13), local numbering with RCM-internal (Fig. 14) and local numbering with *RCM-entire* (Fig. 15).

Generally speaking, *global numbering*, *RCM-internal* and *RCM-entire* provide superior convergence to that of *initial local numbering*. *RCM-entire* attains the best performance and robustness, while *Global numbering* and *RCM-internal* are competitive from the view point of the number of iterations required for convergence (Fig. 21b). The computational cost of *RCM-internal* is, however, slightly more expensive than *global numbering (*Fig. 21a), because **BILU(p)** factorization provides more fill-ins in RCM-internal than global numbering, as shown in Fig. 21c.

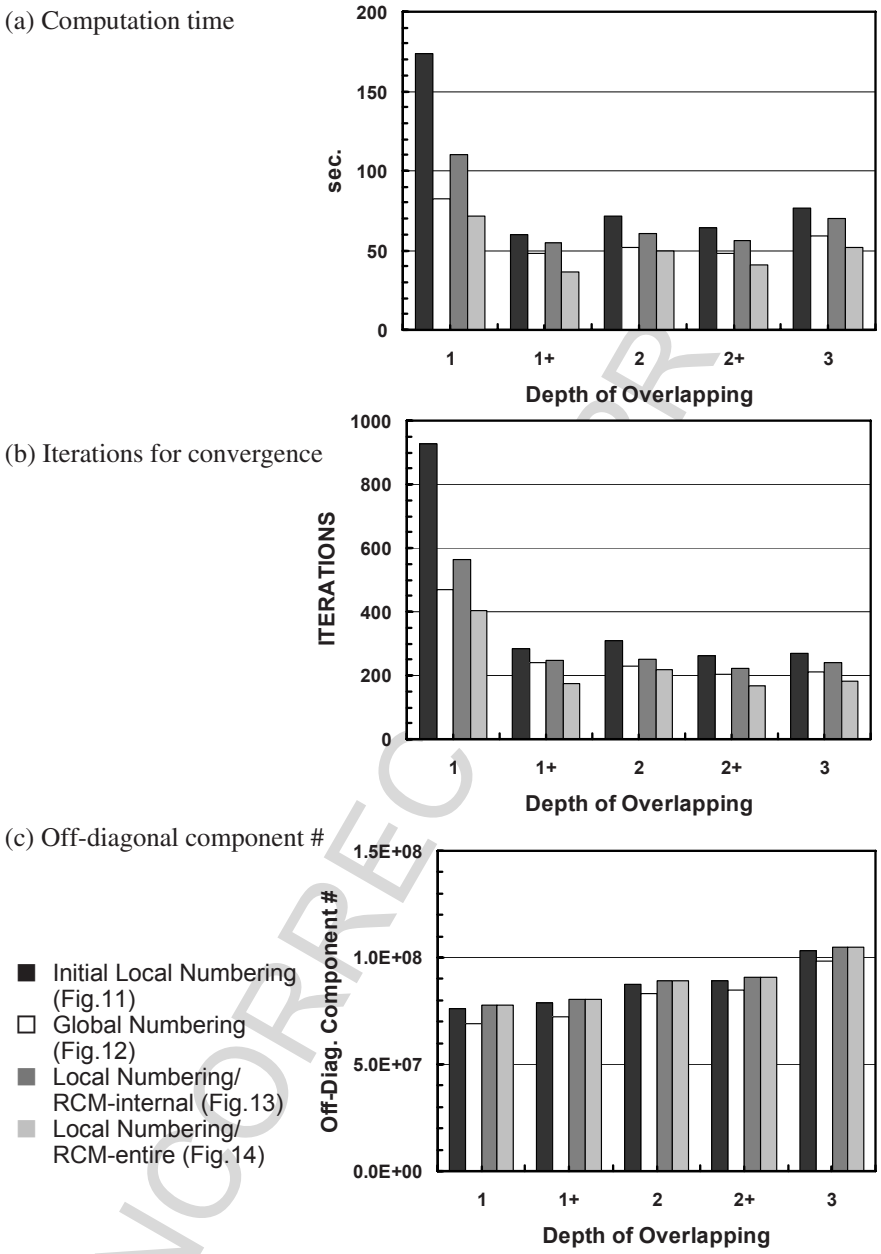


Fig. 21 Effect of overlapping and local reordering for **BILU(1+)** in linear-elastic problem for simple cube geometries of 3,090,903 DOF with contact conditions in Fig. 8 on 64 cores of AMD Opteron 275 cluster with PGI compiler

3.3 Effect of HID

In this section, the robustness and efficiency of HID is compared with that of *selective overlapping*. The plots in Fig. 22 compare the performance of **BILU(1)**, **BILU(1+)**, **BILU(2)** preconditioning for **(d=0)**, **(d=1)** and **(d=1+)** overlapping using local numbering with *RCM-entire* with that of the same preconditioners applied in conjunction with HID.

The depth of overlapping for each local data set provided by HID corresponds to **(d=0)** and is thus in particular smaller than **(d=1)**, as shown in Fig. 19. Figure 22c also shows that cost of HID is competitive with that of **(d=0)**

Figures 22a and b show that **BILU(1)/BILU(1+)/BILU(2)** preconditioners applied with HID are faster and more robust than **BILU(1)/BILU(1+)/BILU(2)-(d=1)**, and are almost competitive with **BILU(1)/BILU(1+)/BILU(2)-(d=1+)**, although **BILU(p)-(d=1+)** is slightly better.

The block structure of the reordered matrix in HID leads to natural parallelism in ILU/IC computations. Thus, HID/PHIDAL-based ILU/IC preconditioners can consider the global effect of external domains in parallel computations. Therefore, although the cost of HID is as cheap as **(d=0)** overlapping, its convergence may be as robust as **(d=1+)** in ill-conditioned problems.

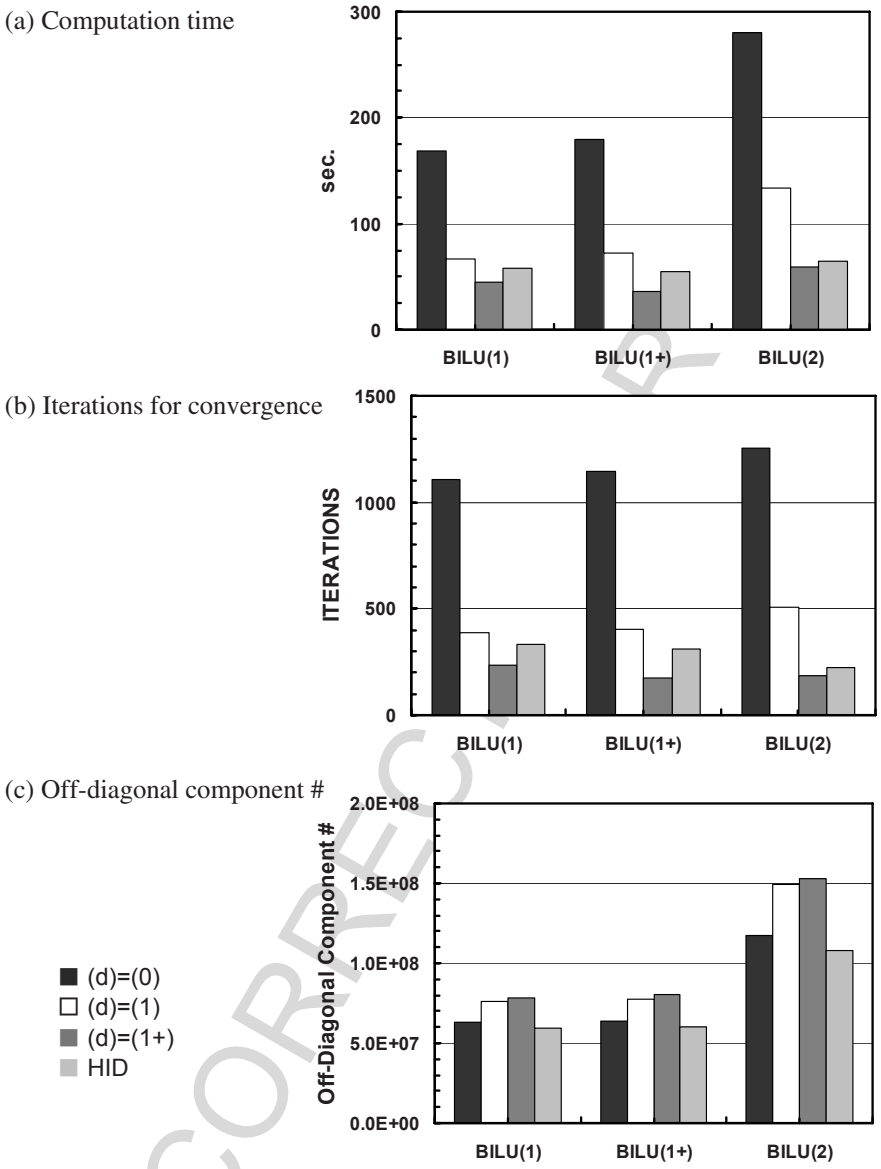


Fig. 22 The effect of overlapping in comparison with HID for the problem in linear elasticity considering simple cube geometries of 3,090,903 DOF with contact conditions as shown in Fig. 8, on 64 cores of AMD Opteron 275 cluster using the PGI compiler, *RCM-entire* reordering applied

4 Examples: Linear-Elastic Problems with Heterogeneous Material Properties

4.1 BILU(p+,ω)-(d+,α)

In (Nakajima 2007a), the **BILU(p)-(d)** method for contact problems was extended to **BILU(p+,ω)-(d+,α)** for ill-conditioned problems with heterogeneous material properties, such as that shown in Fig. 23 (available on accompanying DVD), where **ω** and **α** are threshold parameters for the extension of fill-ins and overlapping.

In applications developed for a heterogeneous distribution of material properties, the coefficient matrices of linear solvers are generally ill-conditioned and the rate of convergence is poor. In **BILU(p+,ω)-(d+,α)**, (p+1)-th-order fill-ins are allowed for pairs of nodes if both nodes are connected to elements for which the Young's modulus is greater than **ω**, while *selective overlapping* is applied to nodes if the nodes are connected to elements for which the Young's modulus is greater than **α**, as shown in Fig. 24.

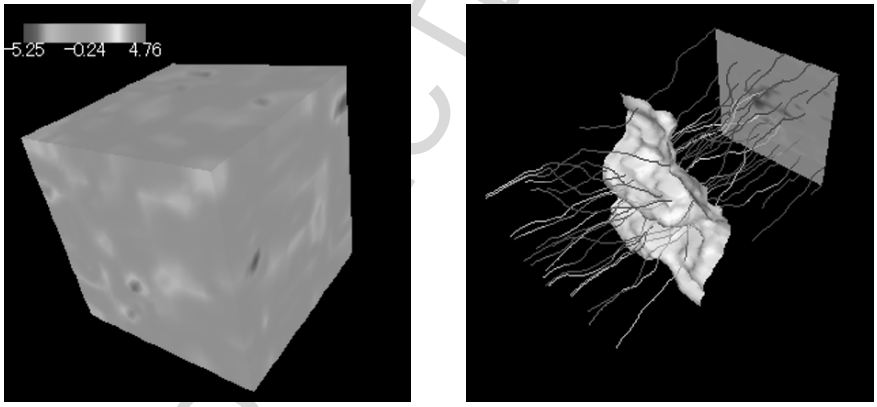


Fig. 23 Heterogeneous distribution of a material property, and groundwater flow through heterogeneous porous media (movie available on accompanying DVD)

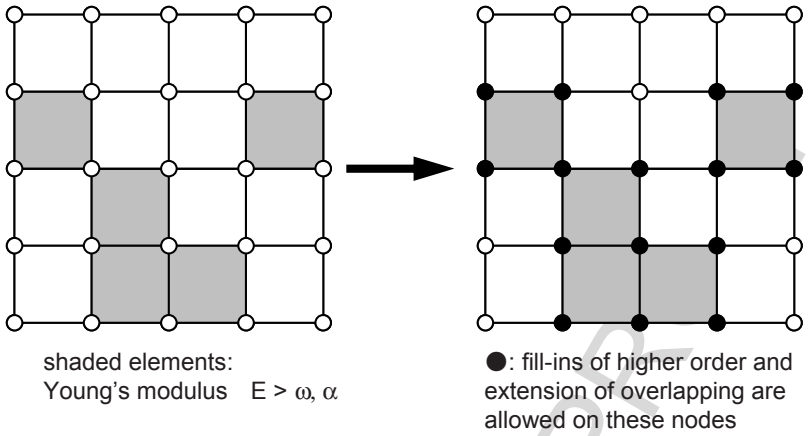


Fig. 24 Selective *fill-ins* and *overlapping* for a heterogeneous field

4.2 Problem Description

Figure 25 describes the boundary conditions for a model problem in linear elasticity considering heterogeneous material of simple cubic geometry. Each element is a cubic tri-linear type finite-element. Poisson's ratio is set to 0.25 for all elements, while the heterogeneous distribution of Young's modulus in each tri-linear element is calculated by a sequential Gauss algorithm, which is widely used in the area of geo-statistics (Deutsch and Journel 1998). The minimum and maximum values of Young's modulus are 10^{-3} and 10^{3}, respectively, where the average value is 1.0.

Symmetric boundary conditions are applied to the x = 0 and y = 0 surfaces, and the Dirichlet fixed condition for deformation in the direction of the z-axis is applied at z = 0. Finally, a uniform distributed load in the direction of the z-axis is applied at the $z = Z_{max}$ surface. This problem is linearly elastic, but the coefficient matrices are particularly ill-conditioned.

The GPBi-CG method for general coefficient matrices is used here as the iterative solution technique, although the coefficient matrices of this problems are positive indefinite. Each node has three DOF in each axis in 3D solid mechanics; therefore, **block ILU (BILU)** type preconditioning has been applied.

The plots of Fig. 26 show results obtained from computations using **BILU** preconditioning applied to the linearly elastic model problem shown in Fig. 24, using a single core of AMD Opteron 275 with PGI compiler.

The number of cubic elements is 32,768 (=32^3), where the total problem size is 107,811 DOF.

BILU(0+,ω), in which additional *selective fill-ins* have been applied to **BILU(0)** for nodes connected to special elements (Young's modulus is larger than **ω**), provides robust and efficient convergence. Although the number of non-zero components of the preconditioning matrices associated with methods **BILU(0)** and **BILU(0+,200)** is comparable, the latter is much more robust and efficient. **BILU(1)** provides better convergence than **BILU(0+,ω)**, but it is more expensive.

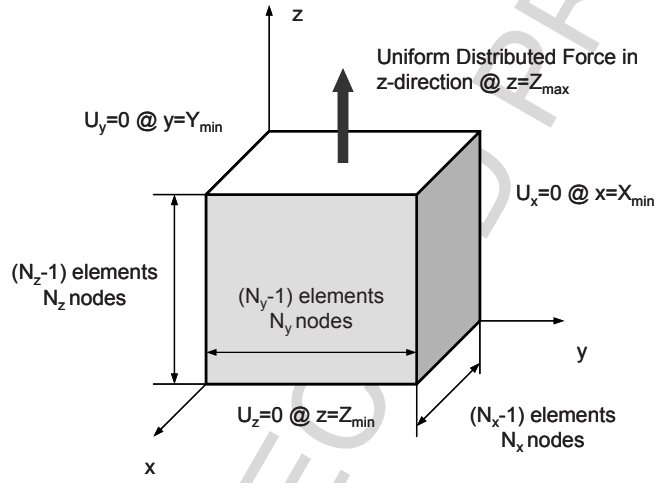
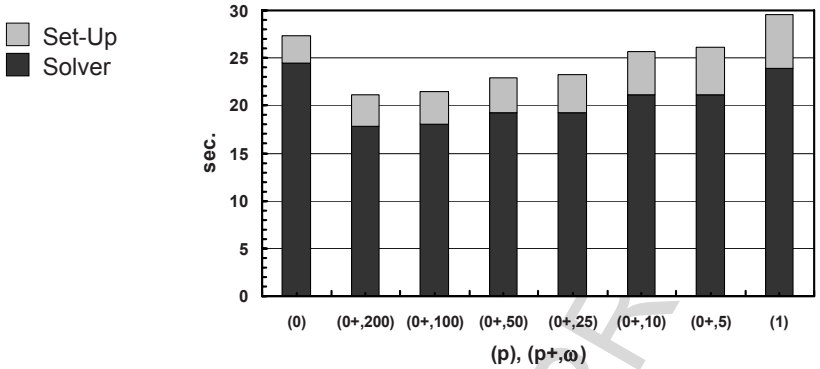
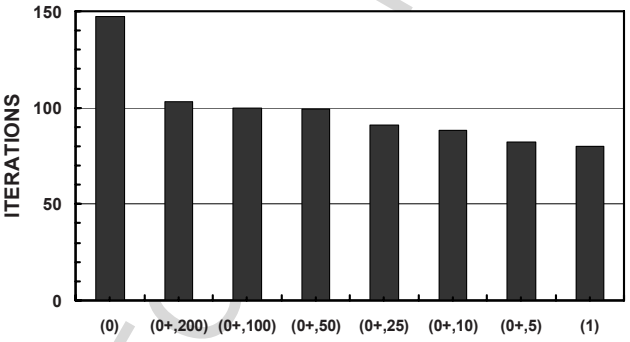


Fig. 25 Boundary conditions of a model problem in linear elasticity considering simple cubic geometries with heterogeneity as shown in Fig. 24

(a) Computation time



(b) Iterations for convergence



(c) Off-diagonal component #

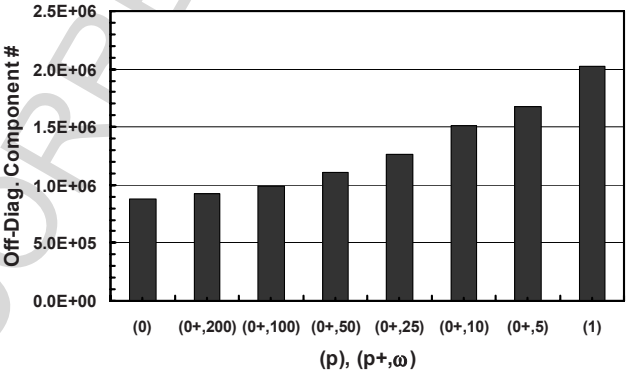


Fig. 26 Results for the problem in linear elasticity considering simple cube geometries of 107,811 DOF with heterogeneity as shown in Fig. 24 on a single core of AMD Opteron 275 using the PGI compiler

4.3 Effect of Selective Fill-Ins and Selective Overlapping

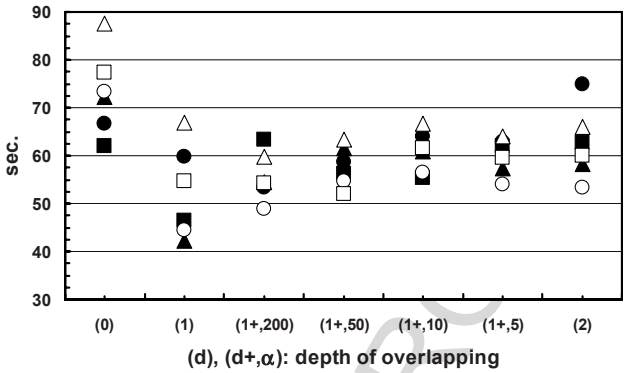
The plots in Fig.27 display the dependence of the results for the heterogeneous test problem in linear elasticity shown in Fig. 24 on the depth of overlapping. All calculations were performed using 64 cores of AMD Opteron 275 cluster with PGI compiler and Pathscale MPI connected through an Infiniband network. The number of cube elements is 1,000,000 ($=100^3$), where the total problem size has 3,090,903 DOF.

Partitioning was applied in an RCB (Simon 1991), and the entire domain has been partitioned into 64 local data sets. Initial local numbering in Fig. 12 has been applied to each local data set.

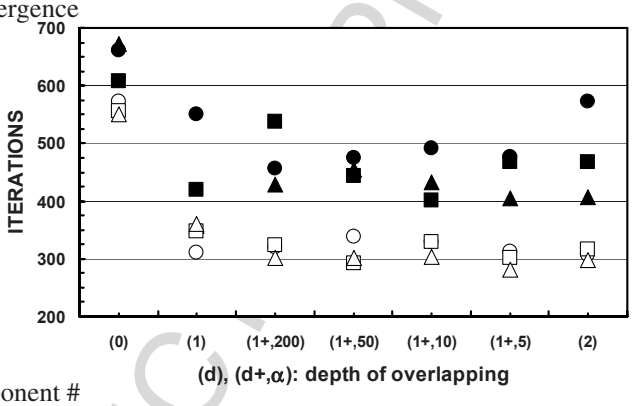
Generally speaking, the convergence rate is improved by the extension of overlapping, but the effect saturates if the depth of overlapping is greater than **(d,α)=(1+,10)**. The effect of selective overlapping is particularly noticeable for increases of the depth of overlapping from **(d=0)** to **(d=1)** or **(d=1+)**, especially for **BILU(1)** and **BILU(0+,ω)**, where **ω** is relatively small.

Generally, amount of computations and communications increases, as the depth of overlapping is larger, but it also saturates if the depth of overlapping is greater than **(d,α)=(1+,10)**, as shown in Fig. 28.

(a) Computation time



(b) Iterations for convergence



(c) Off-diagonal component #

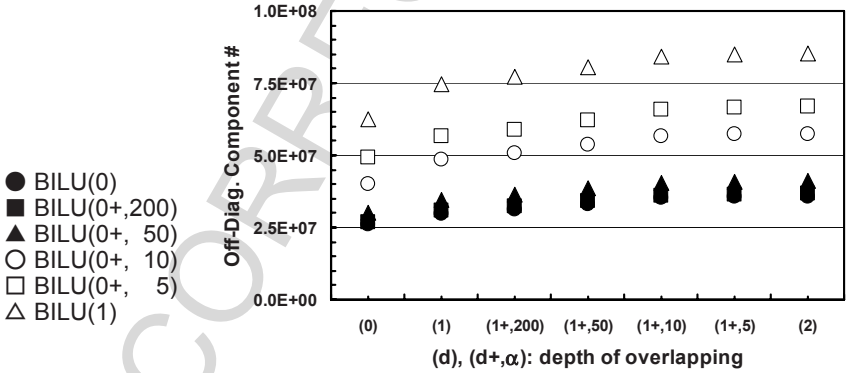


Fig. 27 The effect of overlapping on the results for the problem in linear elasticity considering simple cube geometries of 3,090,903 DOF with heterogeneity as shown in Fig. 24 on 64 cores of an AMD Opteron 275 cluster using the PGI compiler

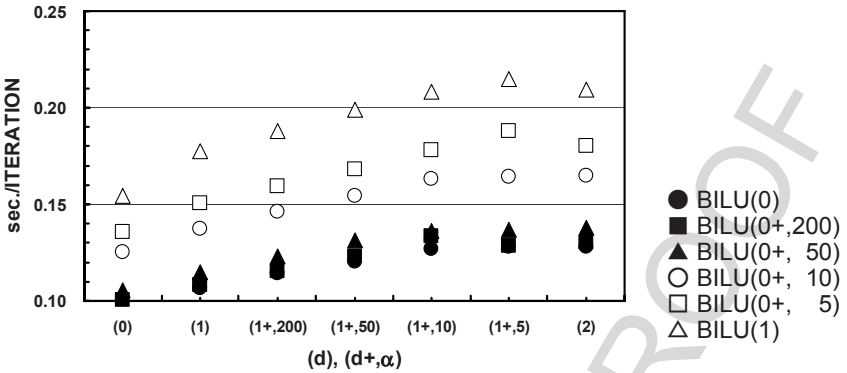


Fig. 28 The averaged elapsed time for each iteration for the problem in linear elasticity considering simple cube geometries of 3,090,903 DOF with heterogeneity as shown in Fig. 24 on 64 cores of an AMD Opteron 275 cluster using the PGI compiler

4.4 Effect of Local Reordering

For the same heterogeneous, linearly elastic test problem considered in the preceding section, the plots in Fig. 29 display the effect of the extension of the depth of overlapping for **BILU(0+,10)**, which was the best performing method in Sect. 4.3, for various types of ordering/numbering schemes.

Generally speaking, the convergence of *global numbering* and *RCM-entire* is much better than that of *initial local numbering*, while *RCM-internal* offers the poorest convergence in every case considered. *Global numbering* and *RCM-entire* offer comparable performance over most of the cases considered here, but *global numbering* is slightly better for larger depths of overlapping.

Figure 30 displays the performance of schemes based on *global numbering* for a variety of preconditioners and depths of overlapping.

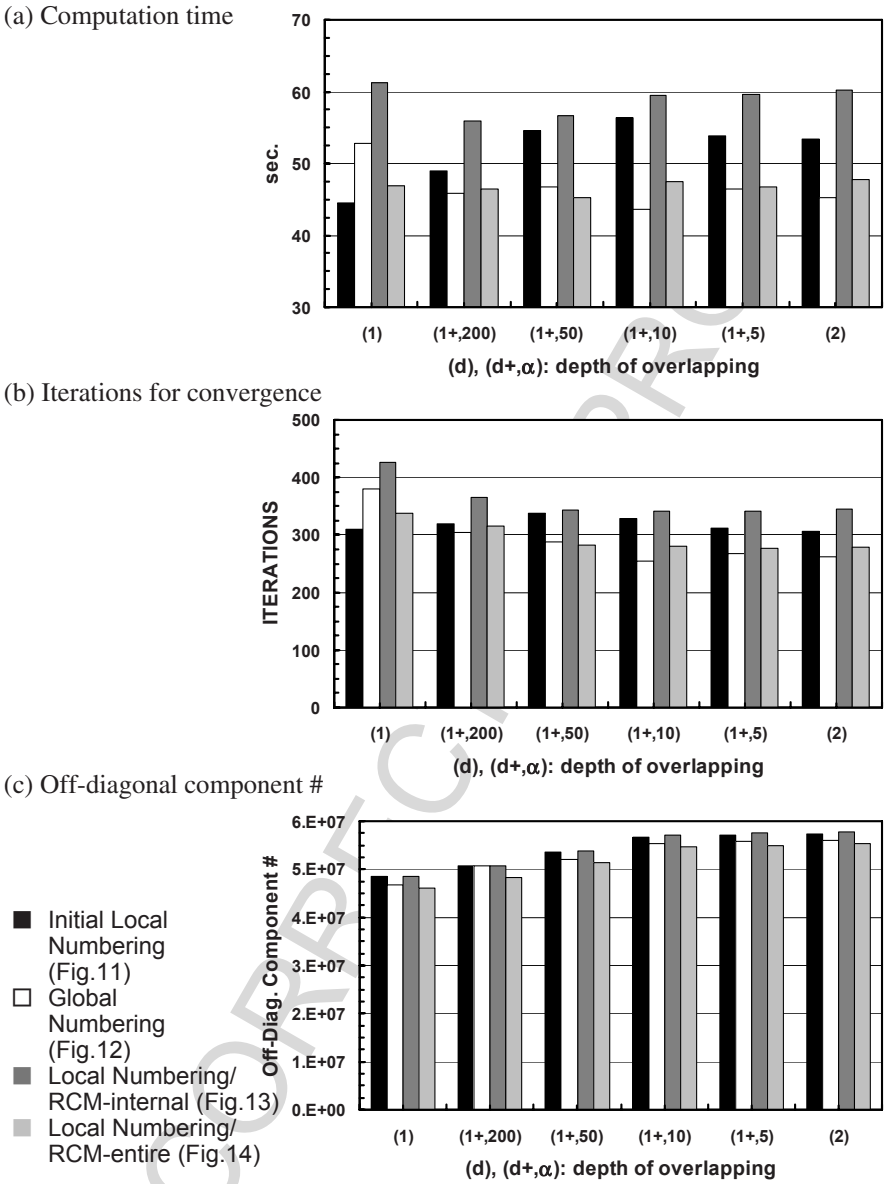
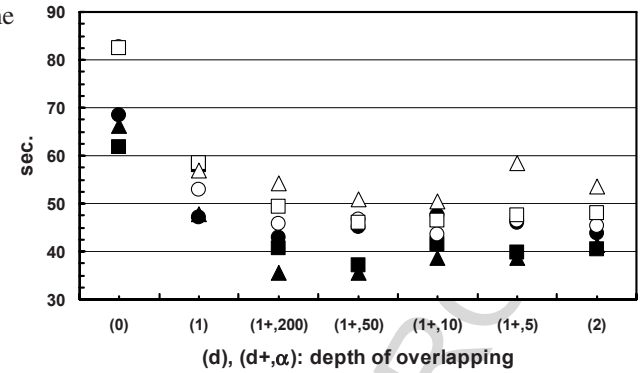
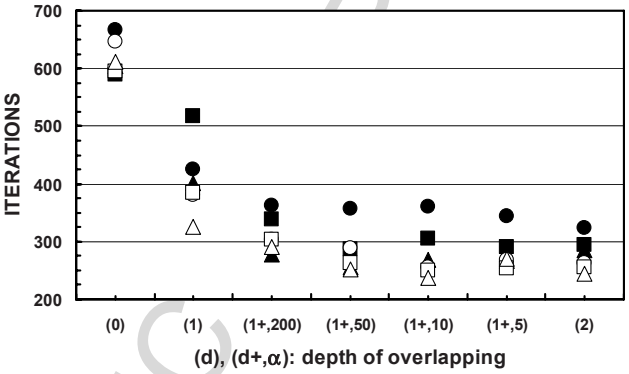


Fig. 29 The effect of overlapping and local reordering on the results for **BILU(0+,10)** applied to the problem in linear elasticity simple cube geometries of 3,090,903 DOF with heterogeneity as shown in Fig. 24 on 64 cores of AMD Opteron 275 cluster using the PGI compiler

(a) Computation time



(b) Iterations for convergence



(c) Off-diagonal component #

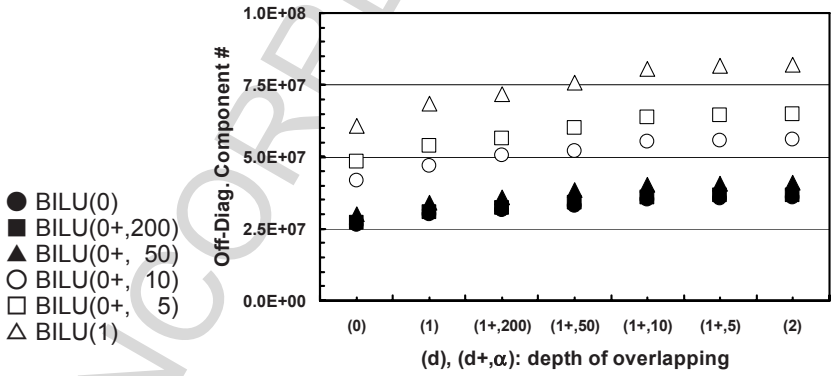


Fig. 30 The effect of overlapping on results for the problem in linear elasticity considering simple cube geometries of 3,090,903 DOF with heterogeneity as shown in Fig. 24 on 64 cores of an AMD Opteron 275 cluster using the PGI compiler, *global numbering* applied

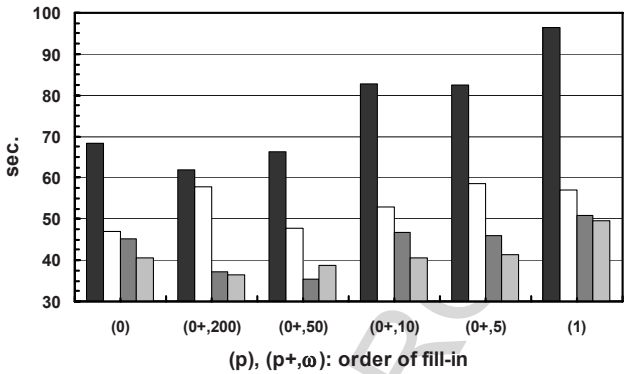
4.5 Effect of HID

In this section the robustness and efficiency of HID is compared with that of *selective overlapping*. The plots in Fig. 31 evaluate the performance of **BILU(0)**, **BILU(0+)**, **BILU(1)** for **(d=0)**, **(d=1)** and **(d=1+,α)** overlapping using *global numbering* (Fig. 13) together with **BILU(1)**, **BILU(1+)**, **BILU(2)** applied with HID.

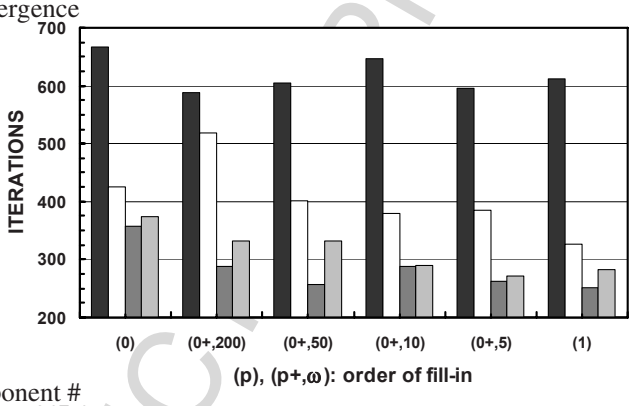
The depth of overlapping for each local data set provided by HID corresponds to **(d=0),** and is thus in particular smaller than **(d=1)**, as shown in Fig. 19. Figure 31c shows that cost of HID is competitive with that of **(d=0)**.

Figure 31a and b show that **BILU(0)/BILU(0+)/BILU(1)** with HID are faster and more robust than **BILU(0)/BILU(0+)/BILU(1)-(d=1)**, and **BILU(0)/BILU(0+)/BILU(1)-(d=1+,α)** in most of the cases considered.

(a) Computation time



(b) Iterations for convergence



(c) Off-diagonal component #

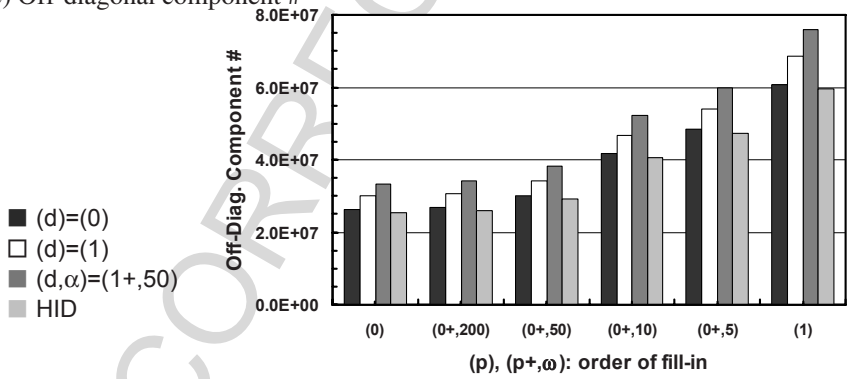


Fig. 31 The effect of overlapping compared with HID on results for the problem in linear elasticity considering simple cube geometries of 3,090,903 DOF with heterogeneity as shown in Fig. 24 on 64 cores of an AMD Opteron 275 cluster using the PGI compiler, *global numbering* applied

5 Concluding Remarks

In this work, the following four approaches have been proposed and introduced as parallel preconditioning methods for ill-conditioned problems:

- Selective fill-ins
- Selective overlapping
- Local reordering
- HID

These methods have been implemented to parallel iterative solvers for finite-element applications, and applied to two types of 3D linear elasticity problems with ill-conditioned coefficient matrices. The first problem includes contact conditions, while the other problem concerns a medium with heterogeneous material properties.

Selective fill-ins and *selective overlapping* are very unique methods, because the dropping rules of the preconditioning matrices are defined according to the properties of individual finite-elements and features of the finite-element applications before assembling entire coefficient matrices.

Generally speaking, **BILU(1+)-(1+)** with selective fill-ins **(p=1+)** and selective overlapping **(d=1+)**, provides the best performance with robustness for contact problems, while **BILU(0+,ω)-(1+,α)** with selective fill-ins **(p=0+)** and selective overlapping **(d=1+)** offers the best performance for heterogeneous cases. The effect of *selective overlapping* is particularly marked for increases in the depth of overlapping from **(d=0)** or **(d=1)** to **(d=1+)**.

In this work, the effect of reordering of local nodes on convergence has also been evaluated. Although the optimum method was different for the two test problems considered here, both *global numbering* and *local numbering with RCM-entire* provide better convergence than the other methods considered. These two methods apply renumbering on both the internal and external nodes in each local data set. If a deeper overlapping of domains is employed in the preconditioning processes, both the internal and external nodes should be reordered for better convergence.

Furthermore, HID was compared with *selective overlapping*. Through reordering the unknowns according to their level numbers, the properties of HID ensure that the coefficient matrix $[A]$ has a block structure. This block structure of the reordered matrix in turn leads to a natural parallelism in ILU/IC computations Thus, HID/PHIDAL-based ILU/IC preconditioners can consider the global effect of external domains in parallel computations. Although HID is as cheap in terms of computational costs as **(d=0)** overlapping, it is as robust as **(d=1+)** and **(d=1+,α)** even for ill-conditioned

problems, as shown in this work. Of the two schemes, HID and *selective overlapping,* it is difficult at this stage to definitively favor one over the other. Further investigation and comparison of these two methods should be undertaken over various types of real applications.

The selection of optimum preconditioning methods with appropriate parameters for parallel computing is a difficult task, especially for ill-conditioned problems, the focus of this work. Usually, there are many parameters to be selected. For example we have order of fill-ins, depth of overlapping, threshold parameters for fill-ins and overlapping, and the method of local reordering in this work. First of all, further investigation of the effect of each parameter on convergence is required for various types of real applications.

There have been some projects considering the automatic selection of preconditioners and parameters, such as the *I-LIB* (*Intelligent Library*) project.[7] They are mainly focusing on the evaluation of the features of coefficient matrices derived from applications. In real applications, convergence of parallel iterative solvers is often affected by local heterogeneity and/or discontinuity of the field, as shown in this paper. Our strategy is to utilize both the global information obtained from derived coefficient matrices and also very local information, such as information obtained from each mesh in finite-element applications.

The strategy of domain decomposition strongly affects the convergence of the method. In this work, it has been shown that optimum methods for reordering of local data differ according to the particular application, although *RCM-entire* generally provides robust convergence. Furthermore, finite-element models for practical simulations contain various sizes and shapes of elements, although only uniform cubic elements were considered in this work.

The first step towards an automatic selection of parameters in parallel preconditioning methods for ill-conditioned problems is the development of an intelligent domain decomposer (partitioner). According to our experiences in this field, convergence declines if domain boundaries are on elements that provide *strong* connections, such as elements with higher values of Young's modulus in heterogeneous cases, and truss elements in contact cases. The intelligent partitioner should also include some rules for the distortion of elements.

If the HID approach is adopted, we do not have to consider the depth of overlapping, but the strategy for domain decomposition is of critical importance, especially for complicated geometries.

[7] http://www.super-computing.org/~kuroda/nadia.html

Acknowledgements

This work is supported by the 21st Century Earth Science COE Program at the University of Tokyo, and CREST/Japan Science and Technology Agency.

References

Barrett R, Berry M, Chan TF, Demmel JW, Donato J, Dongarra JJ, Eijkhout V, Pozo R, Romine C, van der Horst H (1994) Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM

Deutsch CV, Journel AG (1998) GSLIB Geostatistical Software Library and User's Guide, Second Edition, Oxford University Press

Henon P, Saad Y (2007) A Parallel Multistage ILU Factorization Based on a Hierarchical Graph Decomposition. SIAM Journal for Scientific Computing 28, 2266–2293

Iizuka M, Okuda H, Yagawa G (2000) Nonlinear Structural Subsystem of GeoFEM forFault Zone Analysis. Pure and Applied Geophysics 157, 2105–2124

Liou J, Tezduyar TE (1992) Clustered Element-by-Element Computations for Fluid Flow. In Simon HD (ed) Parallel Computational Fluid Dynamics (Implementations and Results), The MIT Press, pp. 167–187

Nakajima K (2003) Parallel Iterative Solvers of GeoFEM with Selective Blocking Preconditioning for Nonlinear Contact Problems on the Earth Simulator, ACM/IEEE Proceedings of SC2003

Nakajima K, Okuda H (2004) Parallel Iterative Solvers for Simulations of Fault Zone Contact using Selective Blocking Reordering. Numerical Linear Algebra with Applications 11, 831–852

Nakajima K (2005) Parallel Preconditioned Iterative Solvers for Contact Problems (in Japanese), Proceedings of Annual Meeting of Japan Society for Applied Mathematics (JSIAM), 18–19

Nakajima K (2007a) 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

Nakajima K (2007b) Parallel Multistage Preconditioners Based on a Hierarchical Graph Decomposition for SMP Cluster Architectures with a Hybrid Parallel Programming Model. Lecture Notes in Computer Science 4782, 384–395

Saad Y (2003) Iterative Methods for Sparse Linear Systems, Second Edition, SIAM

Simon HD (1991) Partitioning of Unstructured Problems for Parallel Processing. Computing Systems in Engineering 2, 135–148

Washio T, Hisada T, Watanabe H, Tezduyar TE (2005) A Robust and Efficient Iterative Linear Solver for Strongly Coupled Fluid-Structure Interaction Problems. Computer Methods in Applied Mechanics and Engineering 194, 4027–4047

Zhang SL (1997) GPBi-CG: Generalized Product-Type Methods Based on Bi-CG for Solving Nonsymmetric Linear Systems. SIAM Journal of Scientific Computing 18, 537–551

6 Appendix 1: Parallel Iterative Solvers in GeoFEM

6.1 Distributed Data Structure

GeoFEM adopts domain decomposition for parallel computing where the entire model is divided into domains, and each domain is assigned to a processing element (PE). A proper definition of the layout of the distributed data structures is an important factor determining the efficiency of parallel computations with unstructured meshes. The local data structures in GeoFEM are node-based with overlapping elements, and as such are appropriate for the preconditioned iterative solvers used in GeoFEM.

Although MPI provides subroutines for communication among processors during computation for structured grids, it is necessary for users to design both the local data structure and communications for unstructured grids. In GeoFEM, the entire region is partitioned in a *node-based* manner and each domain contains the following local data:

- Nodes originally assigned to the domain
- Elements that include the assigned nodes
- All nodes that form elements but are from external domains
- A communication table for sending and receiving data
- Boundary conditions and material properties

Nodes are classified into the following three categories from the viewpoint of message passing:

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

Communication tables between neighboring domains are also included in the local data. Values on *boundary* nodes in the domains are *sent* to the neighboring domains and are *received* as *external* nodes at the *destination* domain. This data structure, described in Fig. 32, and the communication procedure described in Fig. 33 provide excellent parallel efficiency. This type of communication occurs in the procedure for computing the matrix-vector product of Krylov iterative solvers described in the next subsection. The partitioning program in GeoFEM works on a single PE, and divides the initial entire mesh into distributed local data.

In GeoFEM, coefficient matrices for linear solvers are assembled in each domain according to FEM procedures. This process can be performed without communication among processors using the information of overlapping elements.

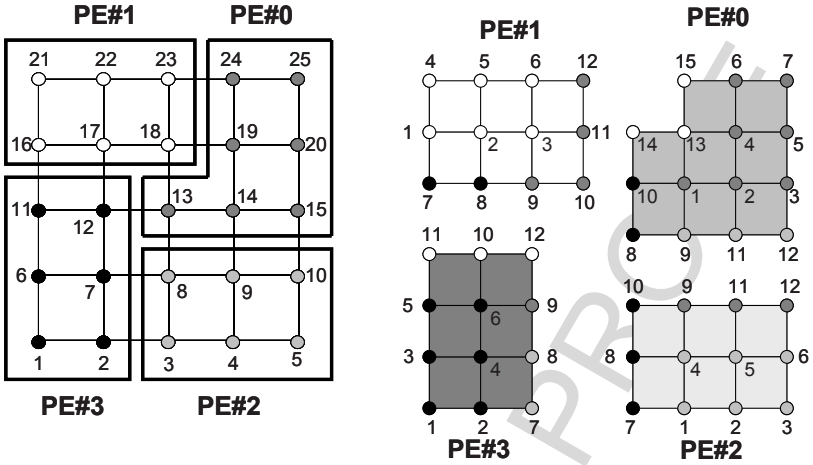


Fig. 32 Node-based partitioning into four PEs

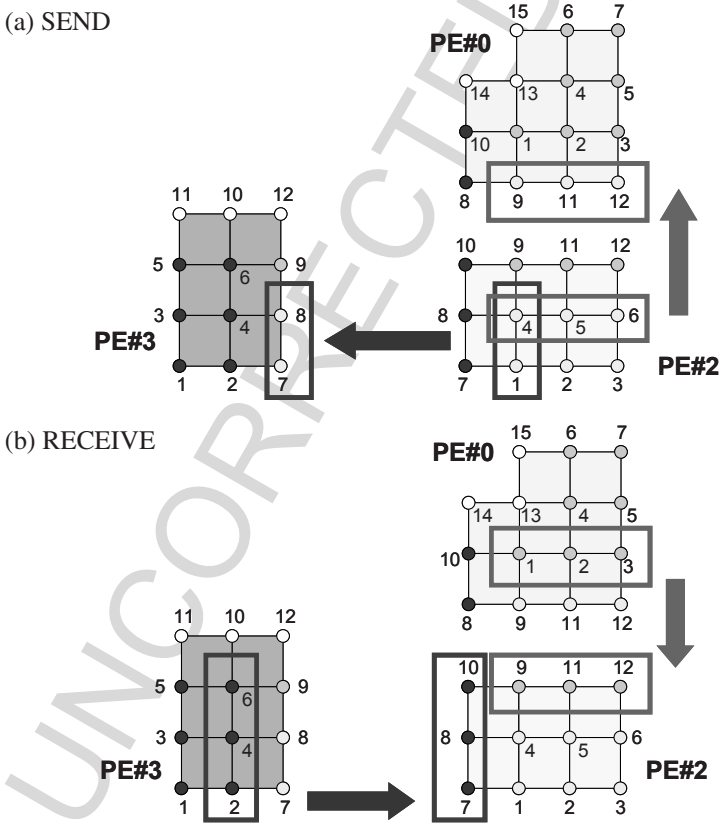


Fig. 33 Communication among processors

6.2 Localized Preconditioning

The incomplete lower-upper (ILU) and incomplete Cholesky (IC) factorization methods are the most popular preconditioning techniques for accelerating the convergence of Krylov iterative methods.

Of the range of ILU preconditioning methods, ILU(0), which does not allow fill-in beyond the original non-zero pattern, is the most commonly used. Backward/forward substitution (BFS) is repeated at each iteration. BFS requires global data dependency, and this type of operation is not suitable for parallel processing in which locality is of utmost importance. Most preconditioned iterative processes are a combination of the following four processes:

- matrix-vector products
- inner dot products
- DAXPY (linear combination of vectors) operations and vector scaling
- preconditioning operations

The first three operations can be parallelized relatively easily. In general, preconditioning operations such as BFS represent almost 50 % of the total computation if ILU(0) is implemented as the preconditioning method. Therefore, a high degree of parallelization is essential for the BFS operation.

The *localized* ILU(0) used in GeoFEM is a *pseudo* ILU(0) preconditioning method that is suitable for parallel processors. This method is not a *global* method, rather, it is a *local* method on each processor or domain. The ILU(0) operation is performed locally for a coefficient matrix assembled on each processor by zeroing out components located outside the processor domain. This is equivalent to solving the problem within each processor with zero Dirichlet boundary conditions during the preconditioning. This *localized* ILU(0) provides data locality on each processor and good parallelization because no inter-processor communications occur during ILU(0) operation. This idea is originally from the incomplete block Jacobi preconditioning method.

However, localized ILU(0) is not as powerful as the global preconditioning method. Generally, the convergence rate degrades as the number of processors and domains increases. At the critical end, if the number of processors is equal to the number of degrees of freedom (DOF), this method performs identically to diagonal scaling.

Table 1 shows the results of a homogeneous solid mechanics example with 3×44^3 DOF solved by the conjugate gradient (CG) method with localized IC(0) preconditioning. Computations were performed on the

Hitachi SR2201, which was operated by the Information Technology Center of the University of Tokyo.[8] Although the number of iterations for convergence increases according to the domain number, this increase is just 30% from 1 to 32 PEs.

Figure 34 shows the work ratio (real computation time/elapsed execution time including communication) for various problem sizes of simple 3D elastic problems with homogeneous boundary conditions. In these computations, the problem size for 1 PE was fixed. The largest case was 196,608,000 DOF on 1024 PEs. Figure 34 shows that the work ratio is higher than 95% if the problem size for 1 PE is sufficiently large. In this case, code was vectorized and a performance of 68.7 GFLOPS was achieved using 1024 PEs. Peak performance of the system was 300 GFLOPS with 1024 PEs; 68.7 GFLOPS corresponds to 22.9% of the peak performance. This good parallel performance is attributed largely to the reduced overhead provided by the use of communication tables as part of the GeoFEM's local data structure.

Table 1 Homogeneous solid mechanics example with 3×44^3 DOF on Hitachi SR2201 solved by CG method with localized IC(0) preconditioning (convergence criteria $\varepsilon=10^{-8}$)

PE #	Iter. #	Sec.	Speed up
1	204	233.7	–
2	253	143.6	1.63
4	259	74.3	3.15
8	264	36.8	6.36
16	262	17.4	13.52
32	268	9.6	24.24
64	274	6.6	35.68

[8] http://www.cc.u-tokyo.ac.jp

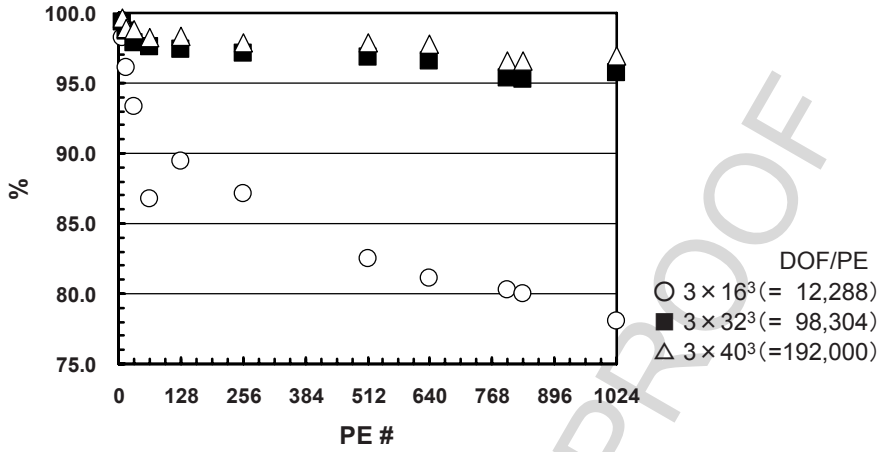


Fig. 34 Parallel performance for various problem sizes for simple 3D elastic solid mechanics on Hitachi SR2201, problem size/PE is fixed, largest case is 196,608,000 DOF on 1024 PEs

7 Appendix 2: Selective Blocking

7.1 Robust Preconditioning Methods for Ill-Conditioned Problems

The IC/ILU factorization methods are the most popular preconditioning techniques for accelerating the convergence of Krylov iterative methods. The typical remedies using an IC/ILU type of preconditioning method for ill-conditioned matrices, which appear in nonlinear simulations using penalty constraints, are as follows:

- Blocking
- Deep Fill-in
- Reordering.

In addition to these methods, a special method called *selective blocking* was also developed for contact problems in Nakajima (2004). In the *selective blocking* method, strongly coupled finite-element nodes in the same contact group coupled through penalty constraints are placed into the same large block (*selective block* or *super* node) and all of the nodes involved are reordered according to this blocking information. Full LU factorization is applied to each selective block. The size of each block is $(3 \times \mathrm{NB}) \times (3 \times \mathrm{NB})$ in 3D problems, where NB is the number of finite-element nodes

in the selective block, which is shown in Fig. 35. Thus, local equations for coupled finite-element nodes in contact groups are solved by means of a *direct* method during preconditioning.

Table 2 shows the convergence of CG solver with various types of preconditioning methods. The linear equations are derived from actual nonlinear contact problems in (Nakajima 2004). By introducing the 3×3 block, the CG solver preconditioned by block IC with no fill-in (i.e., BIC(0)), converges even when λ is as large as 10^6. *Deep fill-in* options provide faster convergence, but the SB-BIC(0) (i.e., BIC(0) preconditioning with *selective blocking* reordering) shows the best performance. SB-BIC(0) usually requires a greater number of iterations for convergence compared to BIC(1) and BIC(2), but the overall performance is better because the computation time for each iteration and set-up is much shorter. As is also shown in Table 2, because no *inter-block* fill-in is considered for SB-BIC(0), the memory requirement for this method is usually as small as that in BIC(0) with no fill-in. Only the *inter-node* fill-in in each *selective block* is considered in SB-BIC(0).

The CG solver with SB-BIC(0) preconditioning can be considered to be a hybrid of iterative and direct methods. Local equations for coupled finite-element nodes in contact groups are solved by means of a direct method during preconditioning. This method combines the efficiency and scalability of iterative methods with the robustness of direct methods.

This idea of selective blocking is also related to the *clustered element-by-element method* (CEBE) (Liou and Tezduyar 1992). In CEBE, elements are partitioned into clusters of elements, with the desired number of elements in each cluster, and the iterations are performed in a cluster-by-cluster fashion. This method is highly suitable for both vectorization and parallelization, if it is used with proper clustering and element grouping schemes. Any number of elements can be brought together to form a cluster, and the number should be viewed as an optimization parameter to minimize computational cost. The CEBE method becomes equivalent to the direct method when the cluster size is equal to the total number of elements. Generally, larger clusters provide better convergence rates because a larger number of fill-in elements are taken into account during factorization, but the cost per iteration cycle increases according to the size of the cluster, as shown in Fig. 36. The trade-off between convergence and computational cost is not clear, but the results of examples by Liou and Tezduyar (1992) show that larger clusters provide better performance.

In *selective blocking*, clusters are formed according to information about the contact groups. Usually, the size of each cluster is much smaller than that in a general CEBE method. If a finite element node does not belong to

any contact groups, it forms a cluster whose size is equal to one in the selective blocking.

Table 2 Iterations/computation time for convergence ($\varepsilon=10^{-8}$) on a single PE of Intel Xeon 2.8 GHz by preconditioned CG for the 3D elastic fault-zone contact problem in (Nakajima 2004) (83,664 DOF), **BIC(p)**: Block IC with p-th-order fill-ins, **SB-BIC(0)**: BIC(0) with the selective blocking reordering

Preconditioning method	λ	Iterations	Set-up (sec.)	Solve (sec.)	Set-up + solve (sec.)	Single iter. (sec.)	Required memory (MB)
Diagonal Scaling	10^2	1,531	<0.01	75.1	75.1	0.049	119
	10^6	N/A	–	–	–	–	
IC(0) (Scalar Type)	10^2	401	0.02	39.2	39.2	0.098	119
	10^6	N/A	–	–	–	–	
BIC(0)	10^2	388	0.02	37.4	37.4	0.097	59
	10^6	2,590	0.01	252.3	252.3	0.097	
BIC(1)	10^2	77	8.5	11.7	20.2	0.152	176
	10^6	78	8.5	11.8	20.3	0.152	
BIC(2)	10^2	59	16.9	13.9	30.8	0.236	319
	10^6	59	16.9	13.9	30.8	0.236	
SB-BIC(0)	10^0	114	0.10	12.9	13.0	0.113	67
	10^6	114	0.10	12.9	13.0	0.113	

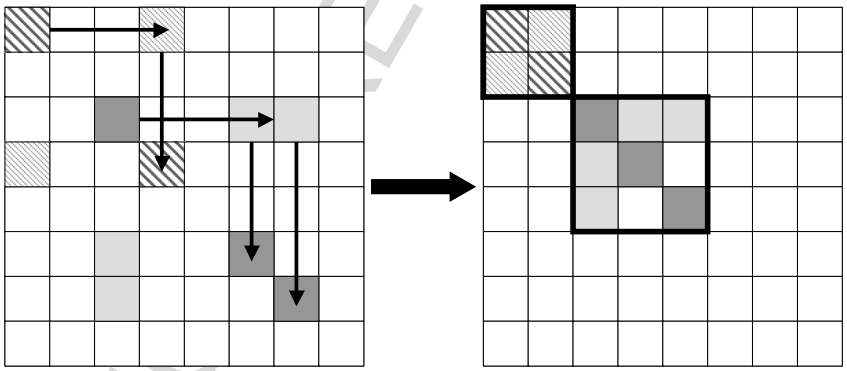


Fig. 35 Procedure of the *selective blocking*, strongly coupled elements are put into the same *selective block*, (**a**) searching for strongly coupled components and (**b**) reordering and selective blocking

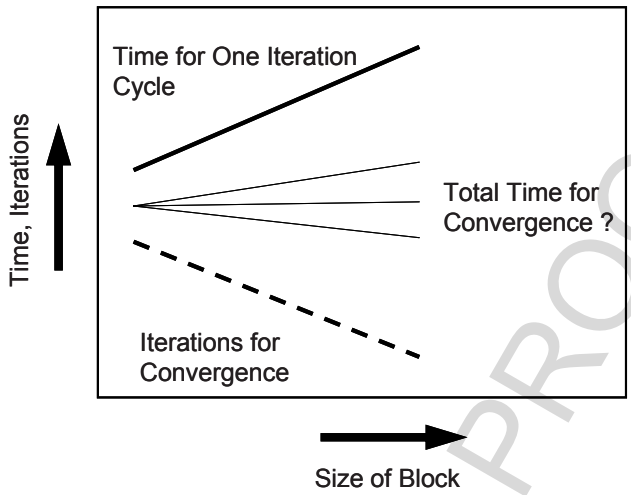


Fig. 36 Trade-off between convergence and computational cost per on iteration cycle according to block size in CEBE type method, based on (Liou and Tezduyar 1992)

In (Nakajima 2003), the robustness of the preconditioning method was estimated according to the eigenvalue distribution of the $[M]^{-1}[A]$ matrix by the method in (Barrett et al. 1994), where $[A]$ is the original coefficient matrix and $[M]^{-1}$ is the inverse of the preconditioning matrix. According to the results, all of the eigenvalues are approximately constant and close to 1.00 for a wide range of λ values except for BIC(0). BIC(1) and BIC(2) provide a slightly better spectral feature than SB-BIC(0).

7.2 Strategy for Parallel Computations

Localized ILU/IC is an efficient parallel preconditioning method, but it is not robust for ill-conditioned problems. Table 3 (left side) shows the results by parallel CG solvers with localized preconditioning on 8 PEs of Intel Xeon 2.8 GHz cluster using distributed matrices, for the problem described in Fig. 1. According to the results, the number of iterations for convergence increases by a factor of 10 in $\lambda=10^6$ cases. This is because the *edge-cuts* occur at inter-domain boundary edges that are included in contact groups.

In order to eliminate these edge-cuts, a partitioning technique has been developed so that all nodes which belong to the same contact group are in the same domain. Moreover, nodes are re-distributed so that

load-balancing among domains should be attained for efficient parallel computing (Fig. 37).

In GeoFEM, there are several types of special elements for contact problems (types 411, 412, 421, 422, 511, 512, 521 and 522). Nodes included in the same elements of these types are connected through penalty constraints and form a contact group. In the new partitioning method, the partitioning process is executed so that these nodes in the same contact elements are on the same domain, or PE. These functions are added to the original domain partitioner in GeoFEM.

Table 3 (right side) shows the results obtained by this partitioning method. The number of iterations for convergence has been dramatically reduced for each preconditioning method although it is larger than that of the single PE cases shown in Table 2 due to localization.

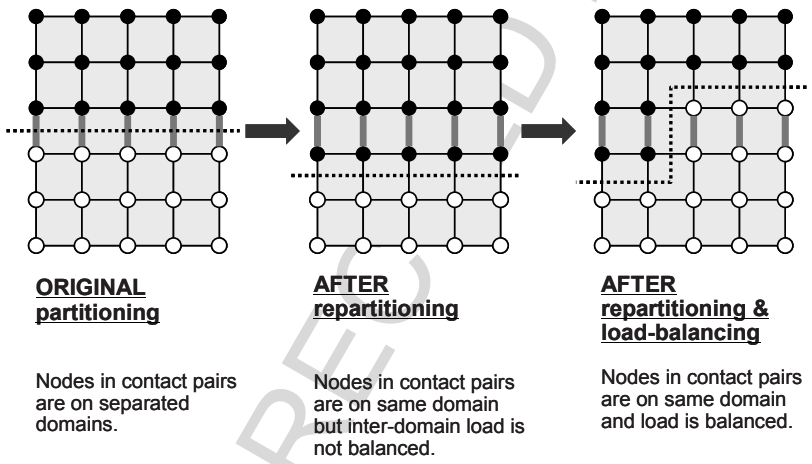


Fig. 37 Partitioning strategy for the nodes in contact groups

Table 3 Iterations/computation time for convergence ($\varepsilon=10^{-8}$) on 8 PEs of Intel Xeon 2.8 GHz cluster by preconditioned CG for the 3D elastic fault-zone contact problem in (Nakajima 2004) (83,664 DOF), **BIC(n)**: Block IC with n-level fill-in, **SB-BIC(0)**: BIC(0) with the selective blocking reordering, effect of repartitioning method in Fig. 37 is evaluated

		ORIGINAL partitioning		**IMPROVED** partitioning	
Preconditioning method	λ	Iterations	Set-up + solve (sec.)	Iterations	Set-up + solve (sec.)
BIC(0)	10^2	703	7.5	489	5.3
	10^6	4,825	50.6	3,477	37.5
BIC(1)	10^2	613	11.3	123	2.7
	10^6	2,701	47.7	123	2.7
BIC(2)	10^2	610	19.5	112	4.7
	10^6	2,448	73.9	112	4.7
SB-BIC(0)	10^0	655	10.9	165	2.9
	10^6	3,498	58.2	166	2.9

7.3 Large-Scale Computations

A large-scale computation was performed on the simple block model with 784,000 elements and 823,813 nodes (Total DOF= 2,471,439) in Fig. 38.

Linear elastic problem on the geometry was solved by parallel iterative solvers using various types of preconditioning methods with the MPC (multiple point constraint) conditions. Domains are partitioned according to the contact group information described in the previous section. Computations were performed using 16–256 PEs on a Hitachi SR2201 at the University of Tokyo.

Table 4 shows the results for various preconditioners. BIC(1), BIC(2) and SB-BIC(0) provide robust convergence but convergence of BIC(0) is very slow. SB-BIC(0) provides the most efficient performance, although the iteration number for convergence is larger than BIC(1) and BIC(2). Figure 39 and Table 4 show the parallel performance for the same problem solved using 16–256 PEs of Hitachi SR2201. BIC(1) and BIC(2) did not work if the PE number was small due to memory limitation. As shown in Table 4 and Fig. 39 the iteration number for convergence increases according to PE number due to the locality of the preconditioning method, but this increase is very slight (only 14% increase from 16 PEs to 256 PEs

for SB-BIC(0)). The speed-up ratio based on elapsed execution time including communication for 256 PEs, is 235 for SB-BIC(0), as extrapolated from the results obtained using 16 PEs.

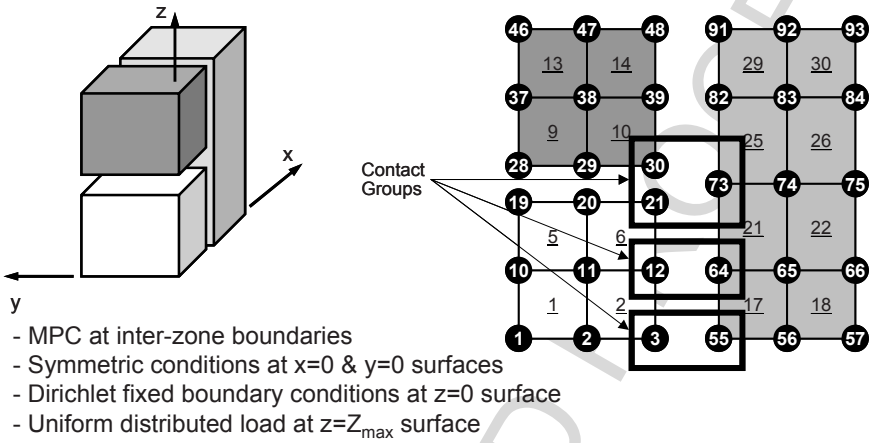


Fig. 38 Description of the simple block model

Table 4 Iterations/elapsed execution time (including factorization, communication overhead) for convergence ($\varepsilon=10^{-8}$) on a Hitachi SR2201 with 256 PEs using preconditioned CG for the 3D elastic contact problem for simple block model with MPC condition in Fig. 38 (2,471,439 DOF), domains are partitioned according to the contact group information, **BIC(p)**: Block IC with p-th-order fill-ins, **SB-BIC(0)**: BIC(0) with the selective blocking reordering

		PE#					
		16	48	96	144	192	256
BIC(0)	iterations	14,459	15,018	15,523	15,820	16,084	16,267
	sec.	13,500	4,810	2,410	1,630	1,270	1,230
	speed-up	16	45	90	133	170	211
BIC(1)	iterations		379	402	424	428	452
	sec.	N/A	236	119	81	62	48
	speed-up		48	95	140	183	236
BIC(2)	iterations			364	387	398	419
	sec.	N/A	N/A	212	140	112	86
	speed-up			96	145	182	217
SB-BIC(0)	iterations	511	527	543	567	569	584
	sec.	555	193	96	64	48	38
	speed-up	16	46	92	139	185	235

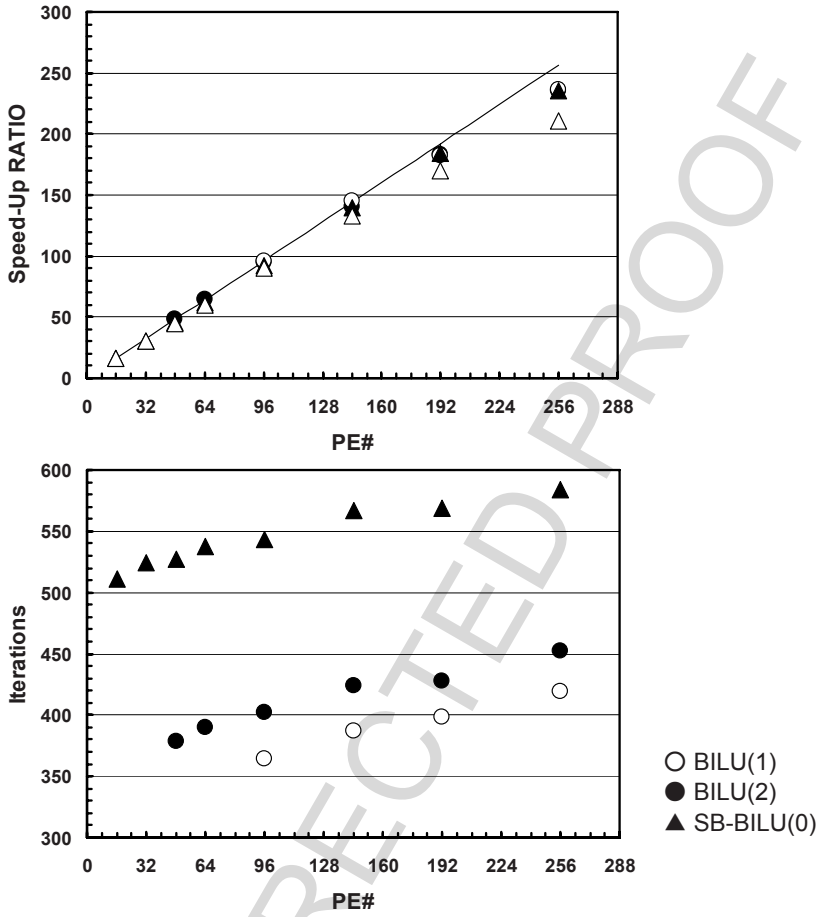


Fig. 39 Parallel performance based on elapsed execution time including communication and iterations for convergence ($\varepsilon=10^{-8}$) on a Hitachi SR2201 with 16–256 PEs using preconditioned CG for the 3D elastic contact problem with MPC condition ($\lambda=10^{6}$) in Fig. 38 (2,471,439 DOF)