Parallel Iterative Solvers for Unstructured Grids using Directive/MPI Hybrid Programming Model for GeoFEM Platform on SMP Cluster Architectures

Kengo Nakajima(1) and Hiroshi Okuda(2)

(1) Department of Computational Earth Sciences, Research Organization for Information Science and Technology (RIST), Tokyo, Japan (e-mail: nakajima@tokyo.rist.or.jp, phone: +81-3-3436-5271, fax: +81-3436-5274) (2) Department of Quantum Engineering and Systems Science, The University of Tokyo, Tokyo, Japan (e-mail: okuda@q.t.u-tokyo.ac.jp, phone: +81-3-5841-7426, fax: +81-3818-3455)

Abstract

In this study, efficient parallel iterative method for unstructured grid has been developed for SMP (shared memory symmetric multiprocessor) cluster architectures on GeoFEM platform. 3-level hybrid parallel programming model has been applied : 1. message passing for inter SMP node communication, 2. loop directive for intra SMP node parallelization and 3. vectorization for each PE (processing element). Simple 3D elastic linear problems with more than 10^8 DOFs have been solved by 3x3 block ICCG(0) with additive Schwartz domain decomposition and PDJDS/CM-RCM ((Parallel Descending-order Jagged Diagonal Storage/Cyclic Multicolor-Reverse Cuthil Mckee)) re-ordering on 16 SMP nodes of Hitachi SR8000 and 20 GFLOPS performance has been obtained. PDJDS/CM-RCM reordering method provides excellent vector and parallel performance in SMP nodes. This re-ordering is essential for parallelization of forward/backward substitution in IC/ILU factorization with global data dependency. Developed method was also tested on NEC SX-4 and attained 969 MFLOPS (48.5% of peak performance) using single processor. Additive Schwartz domain decomposition method provided robustness to the GeoFEM's parallel iterative solvers with localized preconditioning.

1. Introduction

In recent several years, SMP (shared memory symmetric multiprocessor) cluster type architecture has been very popular in massively parallel computers. For example, all of the ASCI machines have adopted this type of architecture[1].

In 1997, the Science and Technology Agency of Japan (STA) began a five-year project to develop a new supercomputer, the Earth Simulator[2]. The goal is the development of both hardware and software for earth science simulations. The Earth Simulator is also a SMP cluster type architecture and consists 640 SMP nodes where each SMP node has 8 vector processors. The present study is conducted as a part of the research on a parallel finite element platform for solid earth simulation, named GeoFEM[3].

In this type of architecture, "Loop Directive + Message Passing" style *hybrid* programming model seems very effective where message passing such as MPI[4] is used in inter SMP node communication and intra SMP node parallelization is guided by loop directive such as OpenMP[5]. A lot of research works have bee done in recent 2 or 3 years[6][7] but most of them are for applications with structured grids such as NPB (NAS Parallel Benchmarks)[8] and very few examples for unstructured grids.

In this study, parallel iterative methods with unstructured grids for SMP cluster architecture have been developed on Hitachi SR8000[9] in the University of Tokyo[10]. Parallel programming model with the following 3 level hierarchy has been developed :

• Inter SMP node	MPI
• Intra SMP node	Compiler Directive for Parallelization
• Individual PE	Compiler Directive for Vectorization/Pseudo Vectorization[9]

Entire domain is partitioned into distributed local data sets[3] and each partition is assigned to one SMP node (Fig.1).

In order to achieve efficient parallel/vector computation for applications with unstructured grids, following 3 issues are critical :

- Local Operation, No Global Dependency
- Continuous Memory Access
- Sufficiently Long Loops

Special re-ordering technique proposed by Washio et. al.[11][12] has been implemented to parallel iterative solvers with localized preconditioning developed in GeoFEM project[3] in order to attain local operation, no global dependency, continuous memory access and sufficiently long loops.

In the following part of this paper, we describe the overview of GeoFEM's parallel iterative solvers, local data structure, re-ordering technique for parallel and vector computation on SMP node and Hitachi SR8000 hardware and show results for 3D solid mechanics.

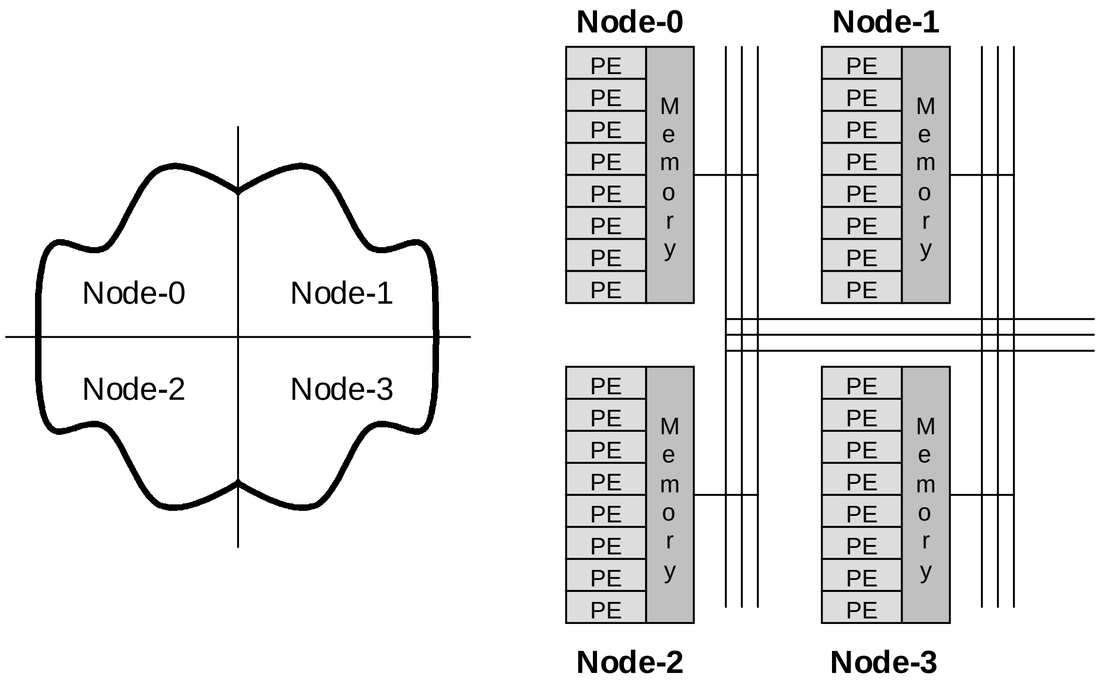


Fig.1 Parallel FEM computation on SMP cluster architecture. Each partition corresponds to SMP node

2. Parallel Iterative Solvers in GeoFEM

2.1 Localized Preconditioning

The Incomplete Lower-Upper (ILU)/Cholesky(IC) factorization method is one of the most popular preconditioning techniques to accelerate the convergence of Krylov subspace iterative methods. Among ILU preconditioners, ILU(0), which allows no fill-in beyond the original non-zero pattern, is the most commonly used. The backward/forward substitution (BFS) is repeated in each iteration. BFS requires global data dependency and is not suitable for parallel processing in which the locality is of utmost importance.

Most preconditioned iterative processes are a combination of :

- Matrix-Vector Products
- Inner Dot Products
- DAXPY ($\alpha\mathbf{x}+\mathbf{y}$) Operations[13] and Vector Scaling
- Preconditioning Operations

The first three operations can be parallelized relatively easily[13]. However, generally speaking, preconditioning (BFS) operation composes almost 50 % of the total computation if ILU(0) is implemented as preconditioner. Therefore, a high degree of parallelization is essential in the BFS operation.

Localized ILU(0) is a *pseudo* ILU(0) preconditioner that is suitable for parallel processors. This method is not a *global* method but a *local* method in each processor or domain. The ILU(0) operation is performed for each processor by zeroing out the matrix components that are located outside the processor domain. This *localized* ILU(0) provides data locality on each processor and good parallelization because no inter-processor communications occur during ILU(0) operation.

However, localized ILU(0) is not as powerful as the global preconditioner. Generally, the convergence rate worsens as the number of processors and domains increases[14][15]. At the critical end, if the processor number is equal to the number of DOF, this method is identical to diagonal scaling.

2.2 Additive Schwartz Domain Decomposition

In order to stabilize the localized ILU(0) preconditioning, additive Schwartz domain decomposition method for overlapped regions[16] has been introduced. Procedures of additive Schwartz method are described as follows :

(1) Global preconditioning process $Mz = r$ where M is preconditioning matrix and r and z are vectors.

(2) If the entire domain is divided into 2 domains Ω_1 and Ω_2, such as in Fig.2(a), preconditioning matrix is solved locally in localized preconditioning :

$$z_{\Omega_1} = M_{\Omega_1}^{-1} r_{\Omega_1}, \quad z_{\Omega_2} = M_{\Omega_2}^{-1} r_{\Omega_2}$$

(3) After local preconditioned matrices are solved, effects of overlapped regions Γ_1 and Γ_2 are introduced by following global nesting correction (Fig.2(b)):

$$z_{\Omega_1}^{n} = z_{\Omega_1}^{n-1} + M_{\Omega_1}^{-1}(r_{\Omega_1} - M_{\Omega_1} z_{\Omega_1}^{n-1} - M_{\Gamma_1} z_{\Gamma_1}^{n-1})$$

$$z_{\Omega_2}^{n} = z_{\Omega_2}^{n-1} + M_{\Omega_2}^{-1}(r_{\Omega_2} - M_{\Omega_2} z_{\Omega_2}^{n-1} - M_{\Gamma_2} z_{\Gamma_2}^{n-1})$$

where "n" denotes cycle number of additive Schwartz domain decomposition

(4) Repeat (2) and (3) until certain convergence

Table 1 shows the effect of additive Schwartz domain decomposition (ASDD) for solid mechanics example with $3x44^3$ DOFs. Computations were done on Hitachi SR2201 in the University of Tokyo and 1 ASDD cycle per iteration has been introduced. Without ASDD, iteration number until convergence increases according to partition number. On the contrary, if ASDD is introduced, iteration number until convergence remains constant although computation time for one iteration increased.

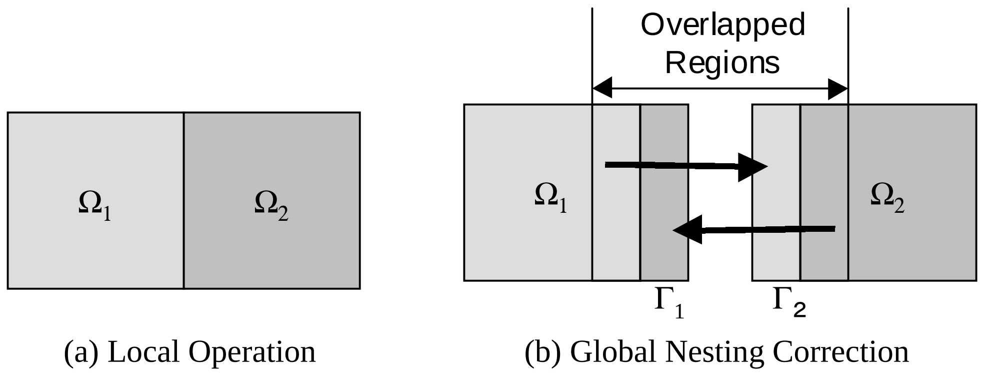


Fig.2 Operations in additive Schwartz domain decomposition : examples for 2 domains

Table 1. Effect of additive Schwartz domain decomposition for solid mechanics example with $3x44^3$ DOFs on Hitachi SR2201.

	NO Additive Schwartz			WITH Additive Schwartz		
PE #	Iter. #	Sec.	Speed Up	Iter.#	Sec.	Speed Up
1	204	233.7	-	144	325.6	-
2	253	143.6	1.63	144	163.1	1.99
4	259	74.3	3.15	145	82.4	3.95
8	264	36.8	6.36	146	39.7	8.21
16	262	17.4	13.52	144	18.7	17.33
32	268	9.6	24.24	147	10.2	31.80
64	274	6.6	35.68	150	6.5	50.07

Number of ASDD cycle/iteration = 1, Convergence Criteria $\varepsilon=10^{-8}$

2.3 Distributed Data Structures

A proper definition of the layout of the distributed data structures is very important for the efficiency of parallel computations with unstructured meshes. GeoFEM's local data structures are node-based with overlapping elements[4][11]. As is mentioned before, each partition is assigned to one SMP node in this study.

Communication among partitions (SMP nodes) occurs during computation. Subroutines for communications in structured finite-difference type grids are provided by MPI. However, users have 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 local data contains the following information :

- Nodes originally assigned to the partition
- Elements which include the assigned nodes
- All nodes which form elements but are located outside of the partition
- 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 nodes)
- External nodes (nodes which form the element in the partition but are located outside of the partition)
- Boundary nodes (*external nodes* of other partitions)

Communication tables between neighboring partitions are also included in the local data. Values in *boundary* nodes in the partitions are *sent* to the neighboring partitions and they are *received* as *external* nodes at the *destination* partition.

This data structure described in Fig.3 provides excellent parallel efficiency[15].

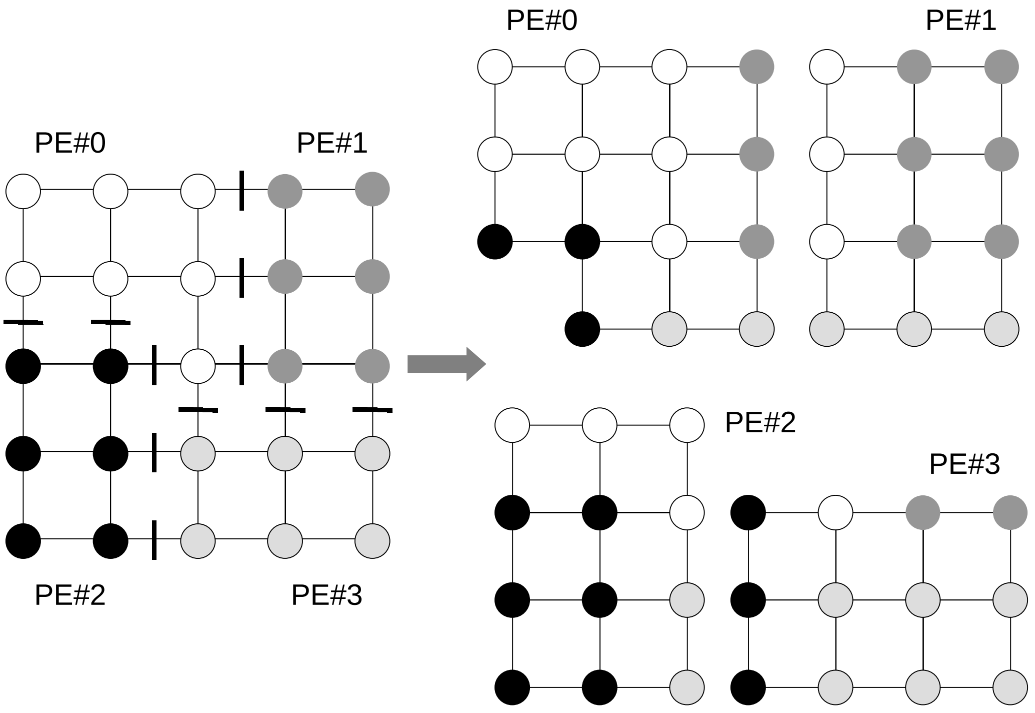


Fig.3 Example of GeoFEM distributed local data structure by node-based partitioning with overlapping elements at partition interfaces

3. Re-Ordering Methods for Parallel/Vector Performance in SMP Node

In this section, re-ordering methods for parallel/vector performance in SMP node are described. As is shown in Fig.1, the entire domain is partitioned into local data sets and each local data corresponds to one SMP node.

3.1 CM-RCM Re-Ordering

As is mentioned before, in order to achieve efficient parallel/vector computation for applications with unstructured grids, following 3 issues are critical :

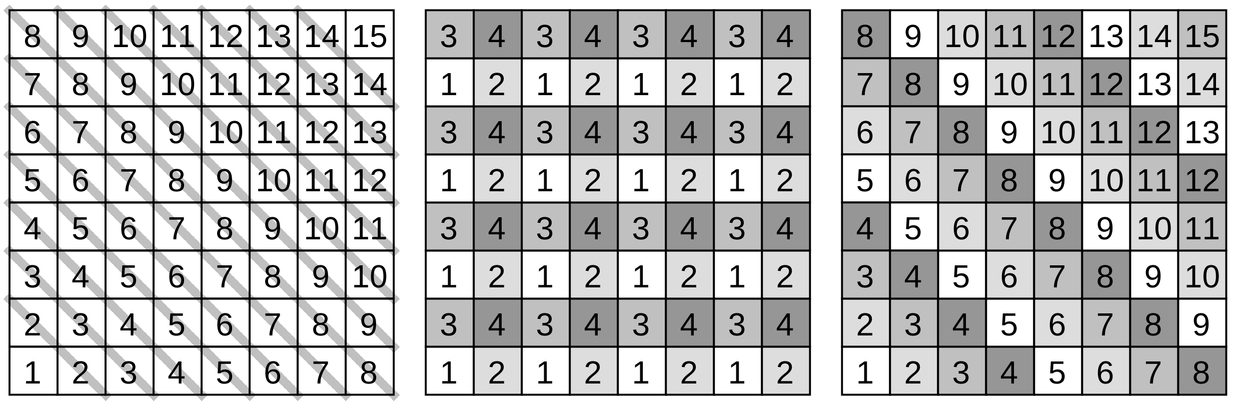
- Local Operation, No Global Dependency
- Continuous Memory Access
- Sufficiently Long Loops

For unstructured grids where data and memory access patterns are very irregular, re-ordering technique is very effective in order to achieve nice parallel and vector performance. Most popular re-ordering methods are Hyperplane/RCM (Reverse Cuthil-Mckee) and Multicolr (MC) [17]. In both methods, elements located on the same hyperplane (or classified in the same color) are independent. Therefore, parallel operation is possible for the elements in the same hyperplane/color and number of elements in the same hyperplane/color should be as large as possible in order to get granularity for parallel computation or sufficiently large loop length for vectorization.

Hyperplane/RCM (Fig.4(a)) re-ordering provides fast convergence of IC/ILU preconditioned Krylov iterative solvers but size of hyperplane is irregular. For example in Fig.4(a), the size of the 1st hyperplane is 1 but 8 for the 8th hyperplane. On the contrary, MC provides uniform element number in each color (Fig.4(b)) but it is widely known that convergence of IC/ILU preconditioned Krylov iterative solvers are rather slow. Convergence can be improved by increasing the number of colors but number of elements in each color decreases.

Remedy for this trade-off between two methods is cyclic multicoloring on Hyperplane/RCM ordering[11]. In this method, hyperplane number is re-numbered in cyclic manner. Fig.4(c) shows the example of CM-RCM (Cyclic Multicolor on Hyperplane/RCM) reordering. In this case, number of color is 4. Therefore 1st, 5th, 9th and 13th hyperplanes in Fig.3(a) are classified in the 1st color. Number of elements in each 4 color is equally 16.

In CM-RCM, number of color should be large enough so that the elements in the same color would be independent from each other.



(a) Hyperplane/RCM (b) Multicolor (MC) : 4 colors (c) CM-RCM : 4 colors

Fig.4 Example of Hyperplane/RCM, Multicolor and CM-RCM re-ordering for 2D geometry

3.2 DJDS Re-Ordering

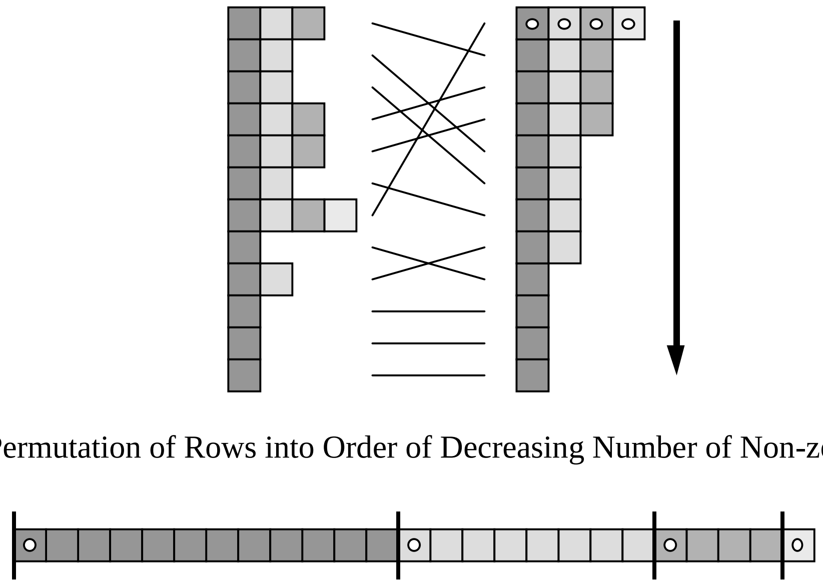
CRS (Compressed Row Storage)[13] type matrix storage format is very memory efficient but length of the innermost loop is relatively small due to matrix-vector operation as follows :

```
do i= 1, N
  do j= 1, NU(i)
    (operations)
    F(i)= F(i) + A(k1)*X(k2)
  enddo
enddo
```

In order to get sufficiently long length of the innermost loop, the following loop exchange is effective :

```
do j= 1, NUmax
  do i= 1, N
    (operations)
    F(i)= F(i) + A(k1)*X(k2)
  enddo
enddo
```

DJDS (Descending-order Jagged Diagonal Storage)[12] is suitable for this type of operation where rows are permuted into order of decreasing number of non-zeros as in Fig.5(a). Because elements on same hyperplane are independent, this permutation inside the hyperplane does not affect the computational results at all. Thus 1D array with continuous memory access for matrix coefficient can be obtained as is shown in Fig.5(b).

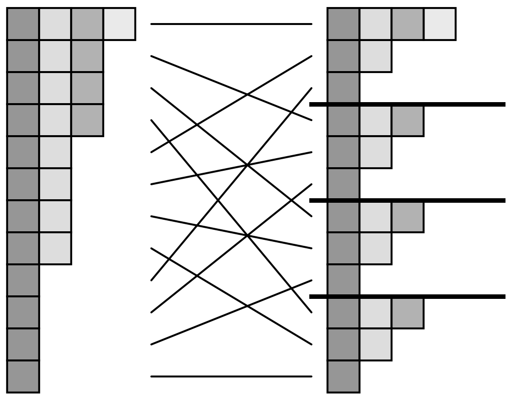

(a) Permutation of Rows into Order of Decreasing Number of Non-zeros

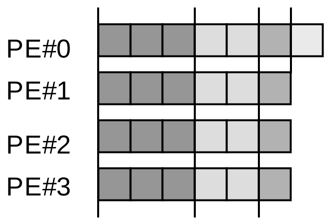
(b) 1D Array for Matrix Coefficient

Fig.5 DJDS (Descending-order Jagged Diagonal Storage) re-ordering for efficient vector/parallel processing

3.3 Distribution in SMP Node : PDJDS Re-Ordering

Currently, we have 1D array for matrix coefficient with continuous memory access. This array is suitable for both parallel and vector computing. The loops for this type of array is easily distributed to each PE in SMP node by loop directive. In order to balance the computational works among PEs in the SMP node, DJDS array should be re-ordered again in cyclic manner. Procedure of this re-ordering (PDJDS = Parallel DJDS) is described in Fig.6 :





(a) (b)

Fig.6 PDJDS (Parallel Descending-order Jagged Diagonal Storage) re-ordering for load-balanced parallel processing in SMP node : Example with 4 PEs in a SMP node (a) Re-ordering in cyclic manner (b) 1D array assigned to each PE after re-ordering and load-balancing

3.4 Summary : Re-Ordering Methods

Procedures of the re-orderings for parallel/vector performance in SMP node described in this section is summarized as follows :

(1) RCM (Reverse Cuthil Mckee) re-ordering on original local matrix for independent sets
(2) CM (Cyclic Multicolor) re-ordering to sufficient and uniform loop length
(3) DJDS (Descending-order Jagged Diagonal Storage) re-ordering for efficient vector processing. 1D arrays for coefficient array with continuous memory access are obtained.
(4) Cyclic re-ordering for load-balancing among PEs on a SMP-node.
(5) PDJDS/CM-RCM re-ordering is completed.

Typical loop structure of the matrix-vector operations is PDJDS /CM-RCM re-ordered matrices is described as follows with pseudo-vector and parallel directives of Hitachi SR8000.

```
do col= 1, COLORtot
  do j= 1, NUmax(col)
*POPTION, INDEP  : Parallelized in SMP node
    do pe= 1, SMP_PE_tot
      iS= NstartU(col,j,pe)
      iE= NendU  (col,j,pe)
*VOPTION, INDEP  : Vecorized for each PE
      do i= iS, iE
        (operations)
      enddo
    enddo
  enddo
enddo
```

4. Hitachi SR8000

Hitachi SR8000 is a parallel system with distributed memory consisting of four to 128 nodes. The nodes are connected by a high-speed multidimensional crossbar network. Each node consists of multiple (8) microprocessors (IPs). These IPs perform high-speed operation simultaneously via the cooperative microprocessor (COMPAS) feature[9].

In this study, the 128-node system in the Computing Center, University of Tokyo has been used. Each node has 8 GFLOPS peak performance and total peak performance is approximately 1 TFLOPS[10].

4.1 Cooperative microprocessors (COMPAS)[9]

This functionality performs high-speed activation of multiple processors in a node simultaneously. Each microprocessor in the node executes one of the threads into which the original program is divided. The compiler automatically performs parallelization in the node so that the user can code data without being aware of hardware. Parallelization of vector operations simplifies conversion from the standard vector operations.

4.2 Pseudo-Vectorization[9]

Pseudo-vectorization performs high-speed numerical computation in the microprocessor. Each microprocessor in a node can pipeline data from memory without stopping subsequent instructions. Therefore, large scale computing is possible at high speed by supplying a large amount of data to the computing element from the memory.

Generally, a RISC processor machine has a cache memory between the processor and the main memory for high-speed data transmission to the processor, thereby increasing the performance. For the numerical calculation programs such as FORTRAN, however, the cache memory cannot be used fully because a large range of array data has to be defined and referenced through loops. Eventually performance decreases.

As a solution to this performance decrease, the SR8000 provides pseudo-vector processing for high-speed transmission of data from the memory to the processor. Pseudo-vector processing generates an object program that processes the data referenced in a loop in one of the following ways.

- The data is loaded beforehand in a floating-point register and the data loading is completed while the loop that references the data is performing calculations from previous iterations. (preload optimizing)
- The data is transferred beforehand onto memory cache and the transfer to the cache memory is completed while the loop that references the data is performing calculations from previous iterations. (prefetch optimizing)

5. Examples

Developed methods are applied to large scale example cases for 3D solid mechanics described in Fig.7 which is linear elastic problem with homogeneous material property and boundary conditions. Each element is a cube with unit edge length. Each node has 3 DOFs (degree of freedoms), therefore there are 3*Nx*Ny*Nz DOFs in the entire problem.

3x3 Block ICCG(0) with PDJDS/MC-RCM re-ordering has been applied where full LU factorization is applied for each 3x3 diagonal block. 1 additive Schwartz domain decomposition is applied to each iteration.

Vector performance has been evaluated on NEC SX-4 (JAERI/CCSE) and Hitachi SR2201 (University of Tokyo) and SMP parallel performance was tested on Hitachi SR8000.

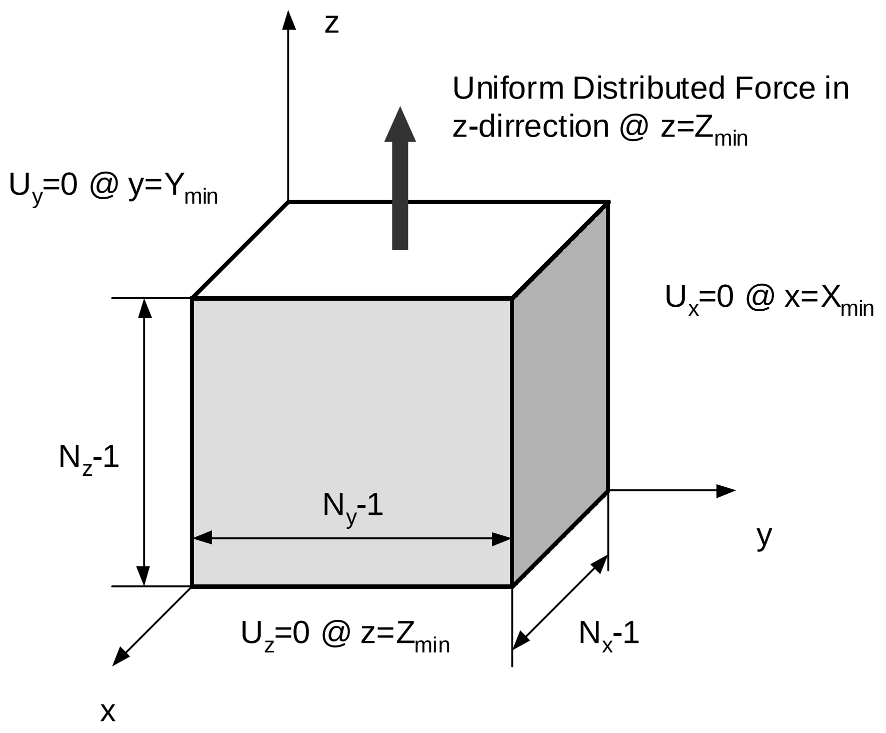


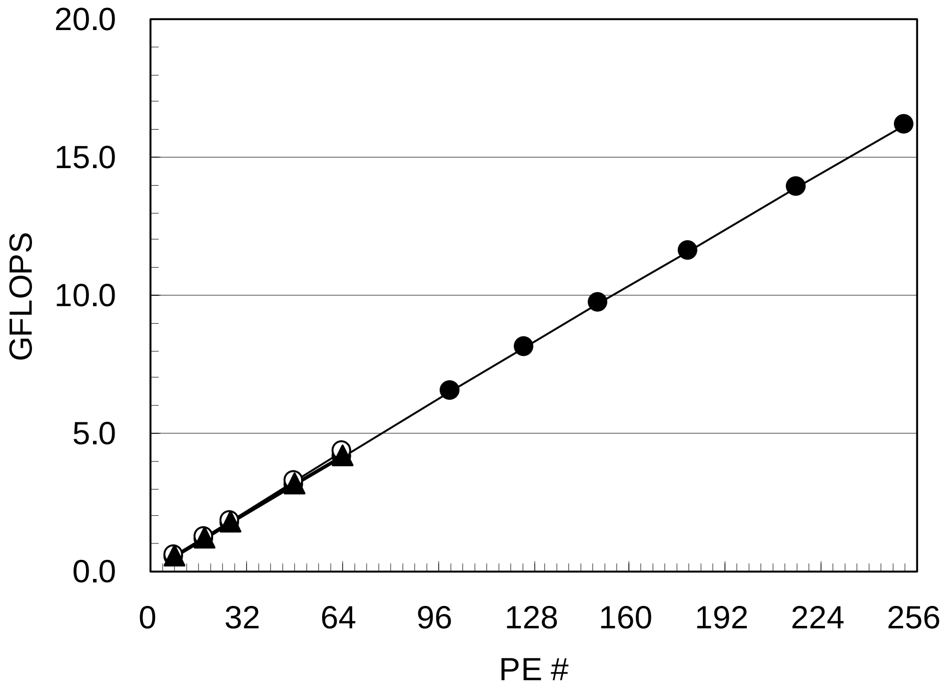
Fig.7 Problem definition and boundary conditions of example cases for 3D solid mechanics. Linear elastic problem with homogeneous material property and boundary conditions. Each element is cube with unit edge length. 3*Nx*Ny*Nz DOFs in the entire problem.

5.1 Vector/Vector Parallel Performance

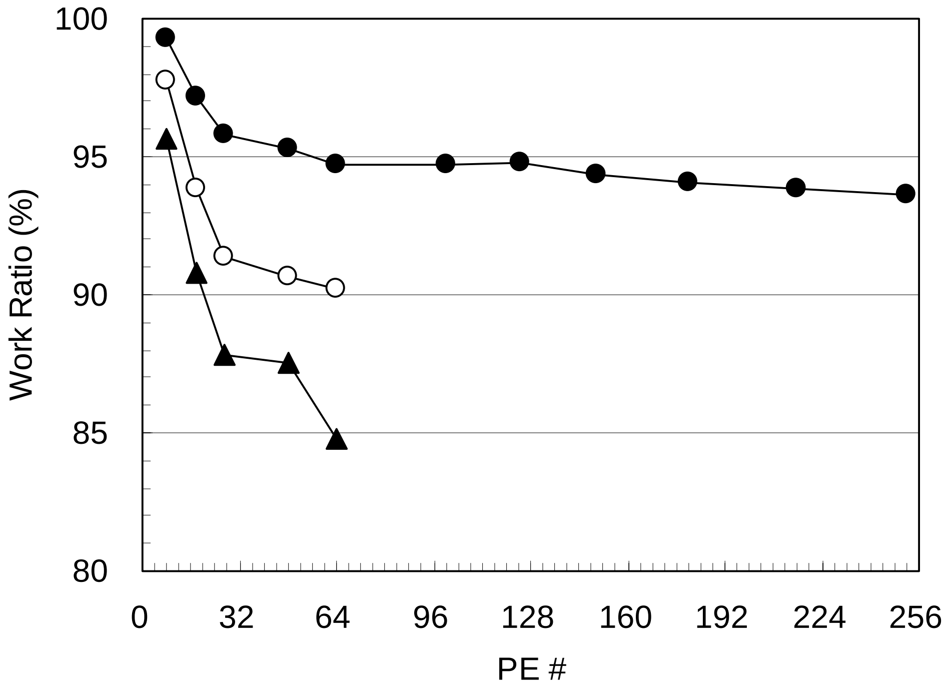
Before computations on Hitachi SR8000, vector performance of the method has been evaluated using NEC SX-4 and Hitachi SR2201. Parallelization for SMP node has not been applied, therefore matrices are re-ordered in DJDS/CM-RCM manner.

Example with 41^3 nodes (= 206,763 DOFs) has been solved on NEC SX-4 using 1 PE. 969 MFLOPS performance was obtained in linear solver part where peak performance was 2 GFLOPS. This corresponds to 48.5% of the peak performance.

Evaluations for parallel computing have been done using Hitachi SR2201 in the University of Tokyo[10]. Pseudo vectorization can be also applied to SR2201 and each PE acts like vector processor. Fig.8 shows GFLOPS rate and work ratio (= real computation time/elapsed execution time including communication) under various problem size configurations. In these computations, problem size for 1 PE has been fixed. The largest case is 27,168,372 DOFs on 252 PEs. Performance of 16.2 GFLOPS has been obtained. Since each processor has 300 MFLOPS peak performance and total peak performance is 75.6 GFLOPS with 252 PEs. Therefore this 16.2 GFLOPS corresponds to 21.4% of the peak performance. Fig.8(b) shows that work ratio is larger than 90% if problem size for 1 PE is sufficiently large, more than 24,000 DOFs in this case.



(a) PE# ~ GFLOPS rate relationship



(b) PE# ~ Work ratio relationship

Fig.8 GFLOPS rate and work ratio (= real computation time/elapsed execution time including communication) under various problem size configurations on Hitachi SR2201. Problem size/PE is fixed. Largest case is 27,168,372 DOFs on 252 PEs with 16.2 GFLOPS (Peak performance = 75.6 GFLOPS), DJDS/CM-RCM re-ordering

●(3x33^3 = 107,811)DOFs /PE ○(3x20^3 = 24,000) ▲(3x15^3 = 10,125)

5.2 SMP Parallel Performance

On Hitachi SR8000, following cases have been carried out :

(1) Speed up for fixed size problem ($3x128^3$ DOFs) from 1 to 16 SMP nodes.
(2) Communication/synchronization overhead for intra SMP node parallellization under various problem size configuration using 1 SMP node
(3) Effect of matrix storage and re-ordering under various problem size configurations using 1 SMP node
(4) Performance evaluation under various problem size configurations using 1-16 SMP nodes

Figure 9 shows the results of (1). In this example, entire problem size has been fixed as $3x128^3$ = 6,291,456 DOFs and speed up rate has bee evaluated for 1-16 SMP nodes. Iteration number until convergence ($\varepsilon=10^{-8}$) is 333(1-node), 337(2-nodes), 338(4-nodes), 341(8-nodes) and 347(16-nodes) respectively. Iteration number remains almost constant even if domain number increases. This is because of additive Schwartz domain decomposition described in 2. In 2, 8 and 8 node cases, "superlinear" occurs. Speed up rate at 16 SMP nodes has been 14.2 (= 88.8%).

Figure 10 shows the results of (2). Communication/synchronization overhead occurs at parallel processing in each SMP node among processors. Work ratio has been measured under various problem size configurations from $3x16^3$=12,288 DOFs to $3x128^3$ = 6,291,456 DOFs on 1 SMP node. Measurement was done by XCLOCK subroutine provided by Hitachi compiler[9]. Results show that overhead is more than 30% in the smallest case but it's less than 10% if the problem size is $3x40^3$=192,000 DOFs which corresponds to 24,000 DOFs/PE and less than 5% for $3x64^3$= 786,432 DOFs (98,304 DOFs/PE) case. According these results, communication/synchronization overhead for intra SMP node is almost negligible in sufficiently large problems.

Figure 11 shows the results of (3), effect of re-ordering. In this case, following 3 types of settings were compared :

- PDJDS/CM-RCM re-ordering
- PDCRS/CM-RCM re-ordering (Parallel Descending-order Compressed Row Storage/CM-RCM)
- CRS without re-ordering

PDCRS/CM-RCM re-ordering is basically identical to PDJDS/CM-RCM but matrices are stored in CRS manner after permutation of rows into order of decreasing number of non-zeros where length of the innermost loop is shorter than PDJDS. Elapsed execution time has been measured under various problem size configurations from $3x16^3$=12,288 DOFs to $3x128^3$ = 6,291,456 DOFs on 1 SMP node.

PDCRS is faster than PDJDS in smaller problems but PDJDS outperforms for larger problems due to the effect of pseudo vectorization. Results of the cases without re-ordering show very poor performance. Parallel computation is impossible for forward/backward substitution (FBS) of the Incomplete Cholesky Factorization process even in the simple geometry solved in this study. This FBS process requires about 50% of the total computation time. If this process is not parallelized, performance is about 20% of the cases with re-ordering. Iteration number until convergence is also larger for cases without re-ordering as is shown in Table 2.

Figure 12 shows the results of (4). This figure corresponds to Fig.8 for results by Hitachi SR2201. Problem size is fixed for one SMP node and node number has been changed from 1 to 16. The largest problem size was 100,663,296 DOFs ($=16x128^3$). Performance is about 20 GFLOPS for 16 SMP node cases and this corresponds to 15.6% of the peak performance.

Fig.12(b) shows that problem size per SMP node is relatively small ($3x32^3$=98,304 DOFs), performance is almost 50% of the results for larger problem size.

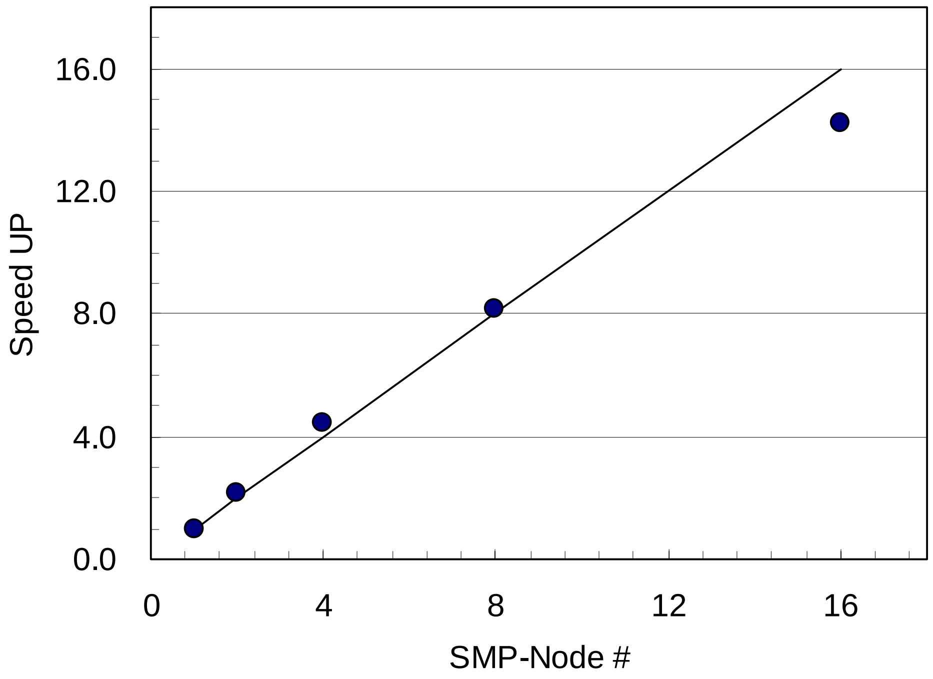


Fig.9 SMP-Node# ~ Speed Up relationship on Hitachi SR8000. Entire problem size is fixed as $3x128^3$= 6,291,456 DOFs. Speed Up rate for 16 SMP-Nodes is 14.2. DJDS/CM-RCM re-ordering.

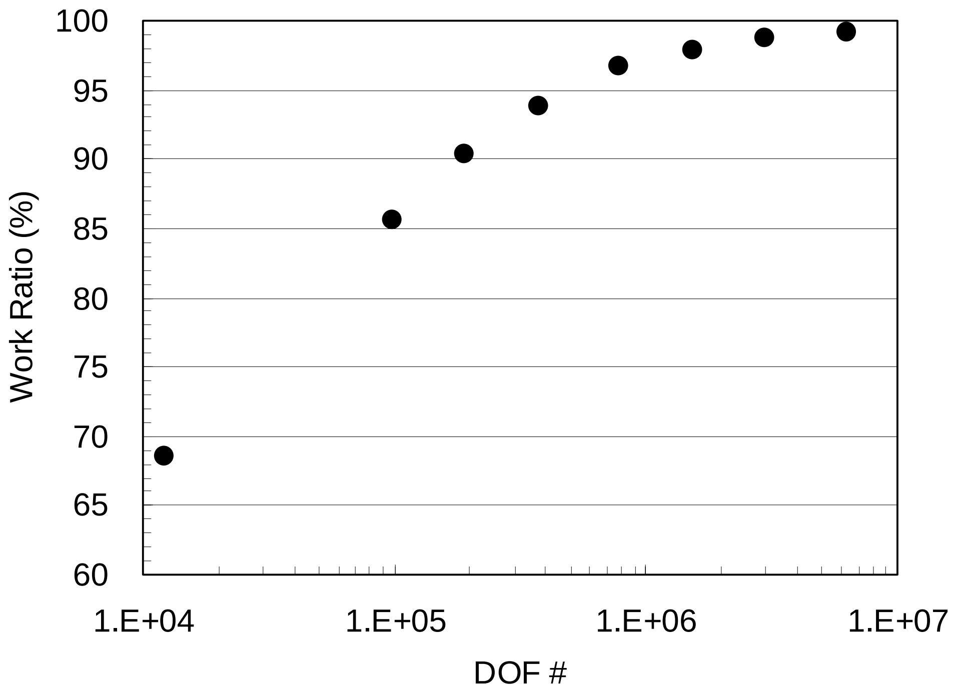


Fig.10 Work ratio (= real computation time/elapsed execution time including communication) under various problem size configurations on Hitachi SR8000 with 1 SMP-Node (= 8 PEs). Work ratio is more than 90% if Problem size is $3x40^3$ DOFs (= 192,000 DOFs) which corresponds to 24,000 DOFs/PE. PDJDS/CM-RCM reordering.

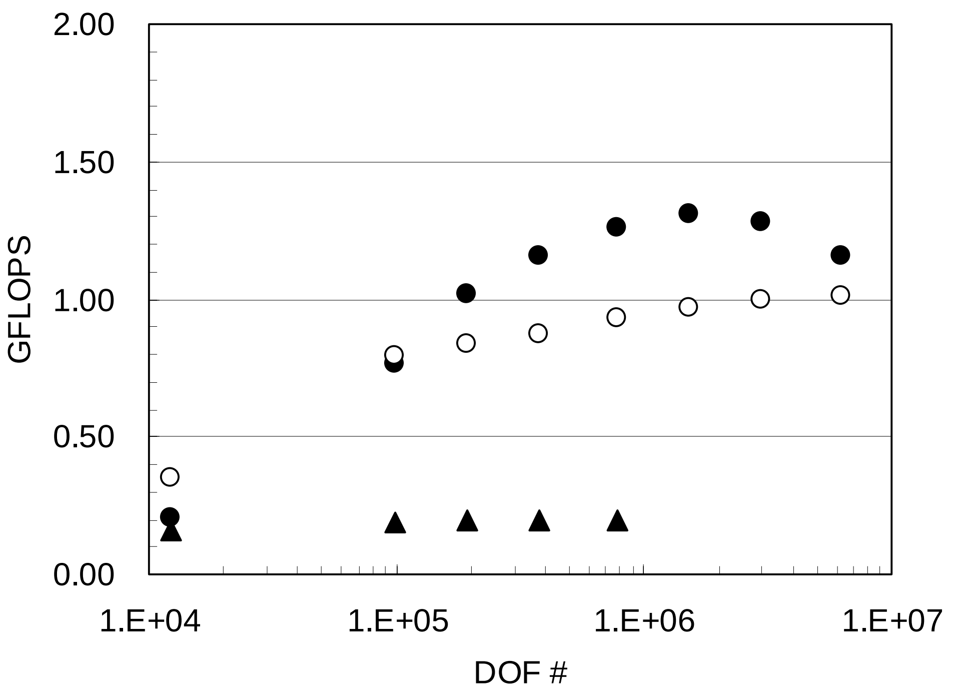
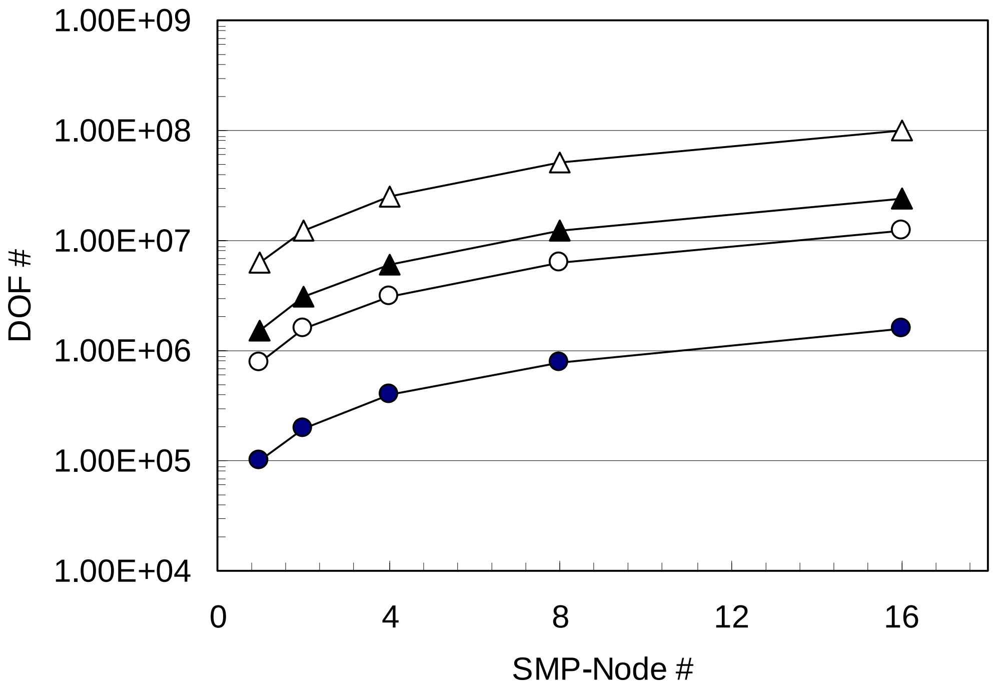


Fig.11 Effect of coefficient matrix storage method and re-ordering under various problem size configurations on Hitachi SR8000 with 1 SMP-Node (= 8 PEs, peak performance= 8.0GFLOPS). Performance of the solver without re-ordering is very bad due to synchronization overhead during forward/backward substitution of the Incomplete Cholesky Factorization. PDCRS/CM-RCM is better than PDJDS/CM-RCM for small problems but getting worse for larger problems because of short length of the innermost loops.

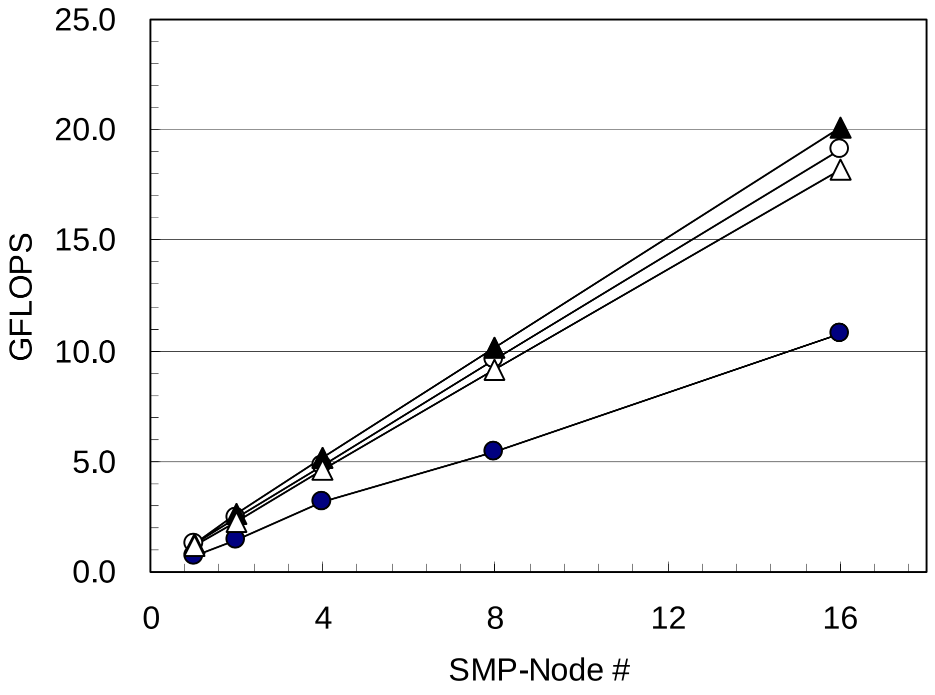
●PDJDS/CM-RCM ○PDCRS/CM-RCM ▲CRS no re-ordering

Table 2. Effect of coefficient matrix storage method and re-ordering under various problem size configurations on Hitachi SR8000 with 1 SMP-Node. Iteration number until convergence $\varepsilon=10^{-8}$

DOF #	With Re-Ordering	Without Re-Ordering
$3x16^3$= 12,288	44	59
$3x32^3$= 98,304	85	116
$3x40^3$= 192,000	106	144
$3x50^3$= 375,000	132	180
$3x40^3$= 786,432	168	230



(a) SMP-Node#~DOF# relationship



(b) SMP-Node#~GFLOPS rate relationship

Fig.12 Problem size and GFLOPS rate under various problem size configurations on Hitachi SR8000. Problem Size/PE is fixed. Largest Case is 100,663,296 DOFs on 16 SMP-Nodes

● $(3x32^3 = 98{,}304)$DOFs /Node ○ $(3x64^3 = 786{,}432)$
▲ $(3x80^3 = 1{,}536{,}000)$ △ $(3x128^3 = 6{,}291{,}456)$

6. Conclusion and Further Study

In this study, efficient parallel iterative method with unstructured grid has been developed for SMP cluster architectures on GeoFEM platform using "Loop Directive+Message Passing" type parallel programming model which contains the following 3 level hierarchy :

• Inter SMP node	MPI
• Intra SMP node	Compiler Directive for Parallelization
• Individual PE	Compiler Directive for Vectorization/Pseudo Vectorization

Simple 3D elastic linear problems with more than 10^8 DOFs have been solved by 3x3 block ICCG(0) with additive Schwartz domain decomposition and PDJDS/CM-RCM re-ordering on 16 SMP nodes of Hitachi SR8000 and 20 GFLOPS performance has been obtained. PDJDS/CM-RCM reordering method provides excellent vector and parallel performance in SMP nodes. Without re-ordering, parallel processing of forward/backward substitution of IC/ILU factorization was impossible due to global data dependency even in the current simple examples. Communication/synchronization overhead in SMP node is less than 10% if the problem size is $3\text{x}40^3$=192,000 DOFs which corresponds to 24,000 DOFs/PE.

Developed method was also tested on NEC SX-4 and attained 969 MFLOPS (48.5% of peak performance) for problem with $2\text{x}10^5$ DOFs using single processor. Vector/parallel efficiency has been evaluated on Hitachi SR2201 with pseudo vectorization under various problem size configurations. The largest case was $2.72\text{x}10^7$ DOFs on 252 PEs with 16.2 GFLOPS (21.4% of the peak performance) has been obtained. Work ratio is larger than 90% if problem size for 1 PE is sufficiently large, more than 24,000 DOFs in this case.

Additive Schwartz domain decomposition method has been implemented to GeoFEM's parallel iterative solvers with localized preconditioning. This method provides robustness to the localized preconditioning and iteration number remains constant even if the number of partition increases for fixed size problems.

In this study, effective hybrid parallel programming model for SMP cluster architecture has been developed but computational performance is not large enough (16% of the peak speed). Therefore we are going to make further optimization, especially for the single PE performance. Other current future plans for this study are as follows :

- Porting the developed method to other SMP cluster type hardware
- Applying the developed method to real-world problems with more complicated geometry

Acknowledgments

This study is a part of the project "Solid Earth Platform for Large Scale Computation" funded by the Ministry of Education, Culture, Sports, Science and Technology, Japan through its "Special Promoting Funds of Science & Technology".

Furthermore the authors would like to thank Professor Yasumasa Kanada (Computing Center, The University of Tokyo) for fruitful discussions on high performance computing, Drs. Shun Doi and Takumi Washio (NEC C&C Research Laboratory) for discussions on preconditioning methods and Mr. Shingo Kudo (Yokohama National University) for his helpful advice on Hitachi SR8000 system.

References

[1] ASCI (Accelerated Strategic Computing Initiative) Web Site : http://www.llnl.gov/asci/

[2] Earth Simulator Research and Development Center Web Site : http://www.gaia.jaeri.go.jp/

[3] GeoFEM Web Site : http://geofem.tokyo.rist.or.jp/

[4] MPI Web Site : http://www.mpi.org

[5] OpenMP Web Site : http//www.openmp.org

[6] Falgout, R. and Jones, J. : "Multigrid on Massively Parallel Architectures", *Sixth European Multigrid Conference*, Ghent, Belgium, September 27-30, 1999.

[7] Cappelo, F. and Etiemble, D. :"MPI versus MPI+OpenMP on the IBM SP for the NAS Benchmarks", *SC2000 Technical Paper*, Dallas, Texas, 2000.

[8] NPB (NAS Parallel Benchmarks) Web Site : http://www.nas.nasa.gov/Research/Software/swdescription.html#NPB

[9] Hitachi SR8000 Web Site : http://www.hitachi.co.jp/Prod/comp/hpc/foruser/sr8000/

[10] Computing Center, The University of Tokyo Web Site : http://www.cc.u-tokyo.ac.jp/

[11] Washio, T., Maruyama, K., Osoda, T., Shimizu, F. and Doi, S. : "Blocking and reordering to achieve highly parallel robust ILU preconditioners", *RIKEN Symposium on Linear Algebra and its Applications*, The Institute of Physical and Chemical Research, 1999, pp.42-49.

[12] Washio, T., Maruyama, K., Osoda, T., Shimizu, F. and Doi, S. : "Efficient implementations of block sparse matrix operations on shared memory vector machines", *SNA2000 : The Fourth International Conference on Supercomputing in Nuclear Applications*, 2000.

[13] Barrett, R., Bery, M., Chan, T.F., Donato, J., Dongarra, J.J., Eijkhout, V., Pozo, R., Romine, C. and van der Vorst, H. : *Templates for the Solution of Linear Systems : Building Blocks for Iterative Methods*, SIAM, 1994.

[14] Nakajima, K., Okuda, H. : "Parallel Iterative Solvers with Localized ILU Preconditioning for Unstructured Grids on Workstation Clusters", *International Journal for Computational Fluid Dynamics* 12 (1999) pp.315-322

[15] Garatani, K., Nakamura, H., Okuda, H., Yagawa, G. : "GeoFEM : High Performance Parallel FEM for Solid Earth", *HPCN Europe 1999*, Amsterdam, The Netherlands, *Lecture Notes in Computer Science* 1593 (1999) pp.133-140

[16] Smith, B., Bjorstad, P. and Gropp, W. : *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge Press, 1996.

[17] Saad, Y. : *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, 1996.