Parallel Iterative Solvers for Unstructured Grids using a Directive/MPI Hybrid Programming Model for the 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: +81-3-3712-5321, fax: +81-3-3712-5552) (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: +81-3-3818-3455)

Abstract

In this paper, an efficient parallel iterative method for unstructured grids developed by the authors for shared memory symmetric multiprocessor (SMP) cluster architectures on the GeoFEM platform is presented. The method is based on a 3-level hybrid parallel programming model, including message passing for inter-SMP node communication, loop directives for intra-SMP node parallelization and vectorization for each processing element (PE). Simple 3D elastic linear problems with more than 10^8 DOF have been solved by 3X3 block ICCG(0) with additive Schwarz domain decomposition and PDJDS/CM-RCM reordering on 16 SMP nodes of a Hitachi SR8000 parallel computer, achieving performance of 20 GFLOPS. The PDJDS/CM-RCM reordering method provides excellent vector and parallel performance in SMP nodes, and is essential for parallelization of forward/backward substitution in IC/ILU factorization with global data dependency. The developed method was also tested on an NEC SX-4 and attained 969 MFLOPS (48.5% of peak performance) using a single processor. The additive Schwarz domain decomposition method provides robustness for the GeoFEM parallel iterative solvers with localized preconditioning.

1. Introduction

In recent years, shared memory symmetric multiprocessor (SMP) cluster architecture has become very popular for massively parallel computers. For example, all Accelerated Strategic Computing Initiative (ASCI) machines have adopted this type of architecture[1].

In 1997, the Science and Technology Agency of Japan (now, the Ministry of Education, Culture, Sports, Science and Technology, Japan) began a 5-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 has SMP cluster architecture and consists of 640 SMP nodes, where each SMP node consists of 8 vector processors. The present study was conducted as part of the research toward developing a parallel finite-element platform for solid earth simulation, named GeoFEM[3].

In this architecture, *loop directives + message passing* style *hybrid* programming model appears to be very effective when message passing such as MPI[4] is used in inter-SMP node communication, and when intra-SMP node parallelization is guided by loop directives such as OpenMP directives[5]. A significant amount of research on this issue has been conducted in recent 2 or 3 years[6][7], but most studies have focused on applications involving structured grids such as the NAS Parallel Benchmarks (NPB)[8], with very few examples for unstructured grids.

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

- Inter-SMP node MPI
- Intra-SMP node Compiler directives for parallelization
- Individual PE Compiler directives for vectorization/pseudo vectorization[9]

The 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, the following 3 issues are critical :

- Local operation and no global dependency
- Continuous memory access
- Sufficiently long loops

A special reordering technique proposed by Washio et. al.[11][12] has been integrated with parallel iterative solvers with localized preconditioning developed in the 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 give an overview of GeoFEM's parallel iterative solvers, local data structure, reordering techniques for parallel and vector computation on SMP nodes and the Hitachi SR8000 hardware system, and present the results for an application to 3D solid mechanics.

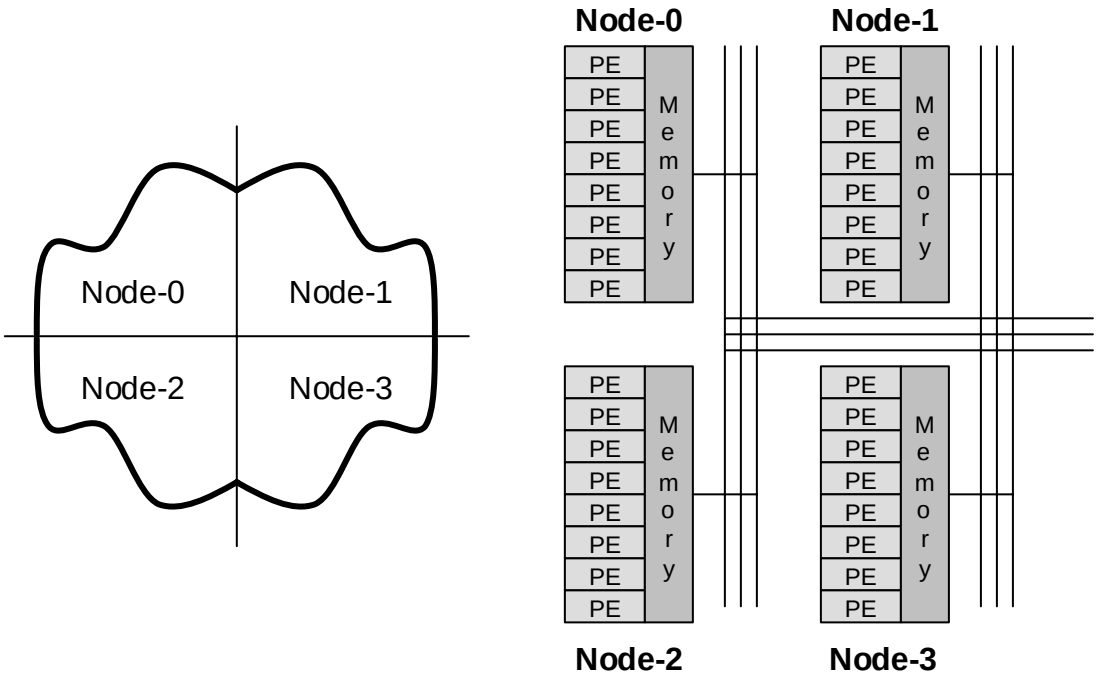


Fig. 1 Parallel FEM computation on SMP cluster architecture.
Each partition corresponds to an SMP node

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 for accelerating the convergence of Krylov iterative methods. Among ILU preconditioners, 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 in each iteration. BFS requires global data dependency, and this type of operation is not suitable for parallel processing in which the locality is of utmost importance.

Most preconditioned iterative processes are a combination of the following :

- Matrix-vector products
- Inner dot products
- DAXPY ($\alpha\mathbf{x}+\mathbf{y}$) operations[13] and vector scaling
- Preconditioning operations

The first 3 operations can be parallelized relatively easily[13]. In general, preconditioning operations (i.e., BFS) represent almost 50 % of the total computation if ILU(0) is implemented as the preconditioner. Therefore, a high degree of parallelization is essential for 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, rather, it is a *local* method on each processor or domain. The ILU(0) operation is performed for each processor by zeroing out matrix components 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 number of processors is equal to the number of the degrees of freedom (DOF), this method performs identically to diagonal scaling.

2.2 Additive Schwarz Domain Decomposition

In order to stabilize localized ILU(0) preconditioning, additive Schwarz domain decomposition (ASDD) for overlapped regions[16] has been introduced. The procedure is as follows :

(1) Global preconditioning $Mz = r$ is performed where M is a 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), the preconditioning matrix is solved locally via localized preconditioning according to :

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

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

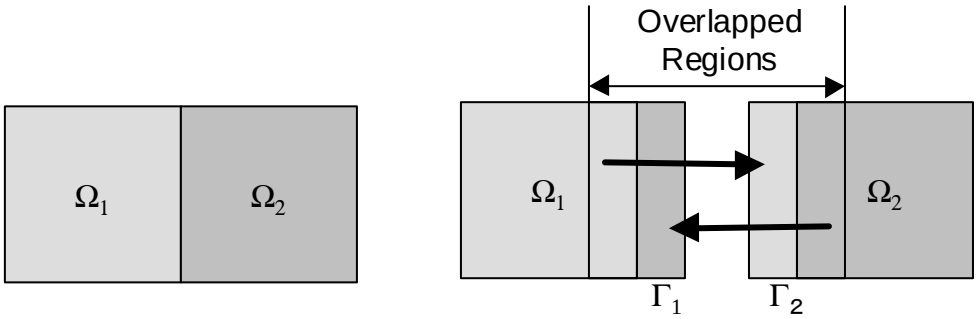
$$z_{\Omega_1}^{n} = z_{\Omega_1}^{n-1} + M_{\Omega_1}^{-1}(r_{\Omega_1} - M_{\Omega_1} z_{\Omega_1}^{n-1} - M_{\Gamma_1} z_{\Gamma_1}^{n-1})$$

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

where n denotes the number of cycles of the additive Schwarz domain decomposition.

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

Table 1 shows the effect of ASDD for a solid mechanics example with 3×44^3 DOF. Computations were performed on a Hitachi SR2201 at the University of Tokyo with 1 ASDD cycle per iteration. Without ASDD, the number of iterations for convergence increases according to the number of partitions. In contrast, when ASDD is introduced, the number of iterations until convergence remains constant, although the computation time for a single iteration increases.



(a) Local operation (b) Global nesting correction

Fig. 2 Operations in ASDD for 2 domains

Table 1. Effect of ASDD for solid mechanics with 3×44^3 DOF on a Hitachi SR2201.

	NO Additive Schwarz			WITH Additive Schwarz		
PE #	Iter. #	Sec.	Speedup	Iter.#	Sec.	Speedup
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 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 grids are provided by MPI. However, users are required 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 partition contains the following local data :

- Nodes originally assigned to the partition
- Elements that include the assigned nodes
- All nodes that form elements but are located outside of the partition
- The communication table for sending and receiving data
- Boundary conditions and material properties

Nodes are classified into the following 3 categories from the viewpoint of message passing :

- Internal nodes (originally assigned nodes)
- External nodes (nodes that 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 on *boundary* nodes in the partitions are *sent* to the neighboring partitions and are *received* as *external* nodes at the *destination* partition.

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

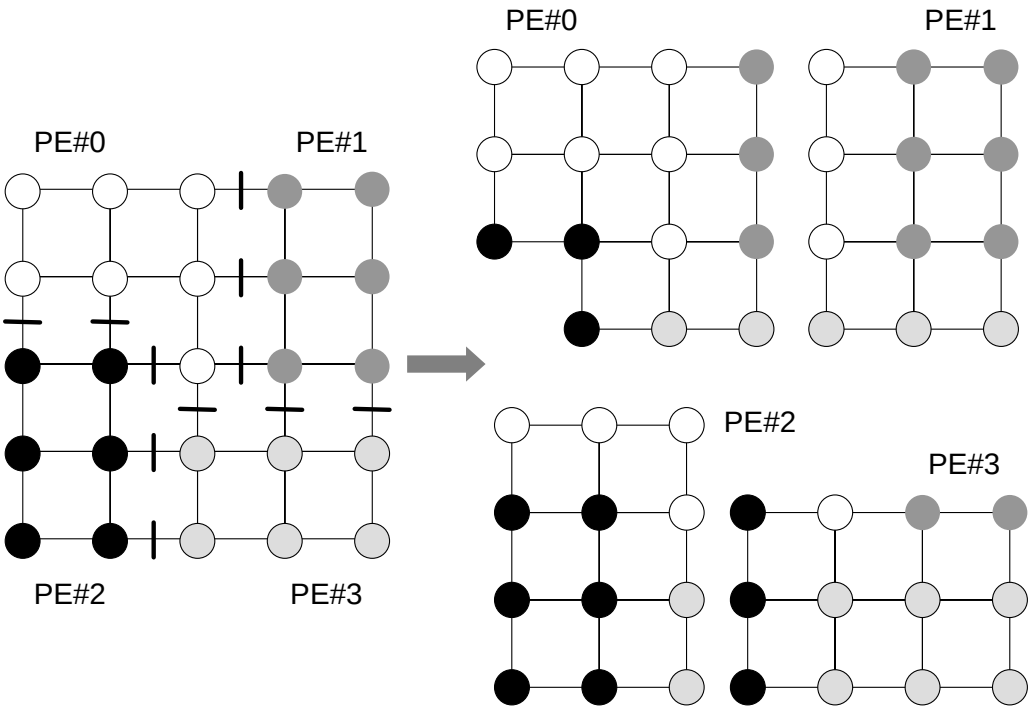


Fig. 3 Example of GeoFEM distributed local data structure by node-based partitioning with overlapping elements at partition interfaces

3. Reordering Methods for Parallel/Vector Performance Using SMP Nodes

As shown in Fig. 1, the entire domain is partitioned into local data sets and each local data set corresponds to one SMP node.

3.1 Cyclic Multicolor – Reverse Cuthil McKee Reordering

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

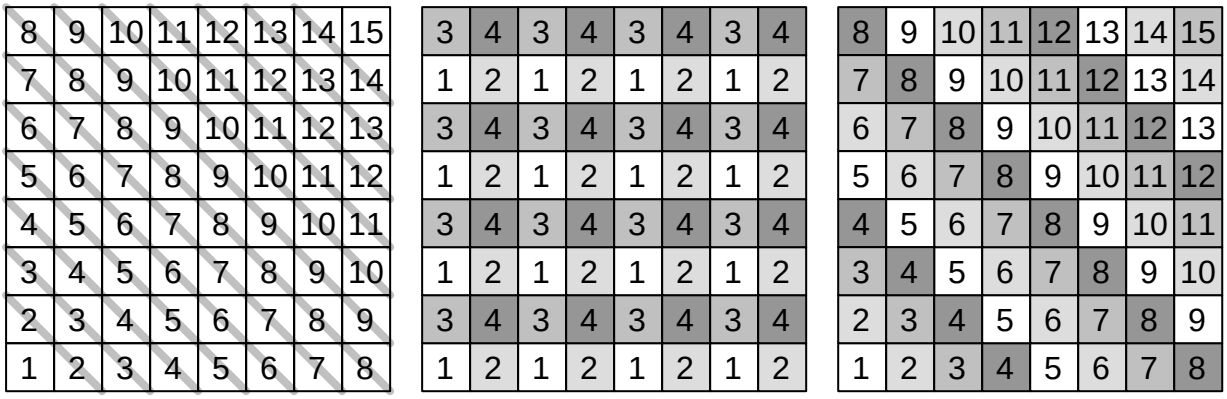
- Local operations and no global dependency
- Continuous memory access
- Sufficiently long loops

For unstructured grids, in which data and memory access patterns are very irregular, the reordering technique is very effective for achieving high parallel and vector performance. The popular reordering methods are hyperplane/reverse Cuthil-McKee (RCM) and multicoloring [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 the number of elements in the same hyperplane/color should be as large as possible in order to obtain high granularity for parallel computation or sufficiently large loop length for vectorization.

Hyperplane/RCM (Fig. 4(a)) reordering provides fast convergence of IC/ILU-preconditioned Krylov iterative solvers, yet with irregular hyperplane size. For example in Fig. 4(a), the 1st hyperplane is of size 1, while the 8th hyperplane is of size 8. In contrast, multicoloring provides a uniform element number in each color (Fig. 4(b)). However, it is widely known that the convergence of IC/ILU-preconditioned Krylov iterative solvers is rather slow. Convergence can be improved by increasing the number of colors, but this reduces the number of elements in each color.

The solution for this trade-off is cyclic multicoloring (CM) on hyperplane/RCM [11]. In this method, the hyperplanes are renumbered in a cyclic manner. Figure 4(c) shows an example of CM-RCM reordering. In this case, there are 4 colors ; the 1st, 5th, 9th and 13th hyperplanes in Fig. 3(a) are classified into the 1st color. There are 16 elements in each color.

In CM-RCM, the number of colors should be large enough to ensure that elements in the same color are independent.



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

Fig. 4 Example of hyperplane/RCM, multicoloring and CM-RCM reordering for 2D geometry

3.2 DJDS Reordering

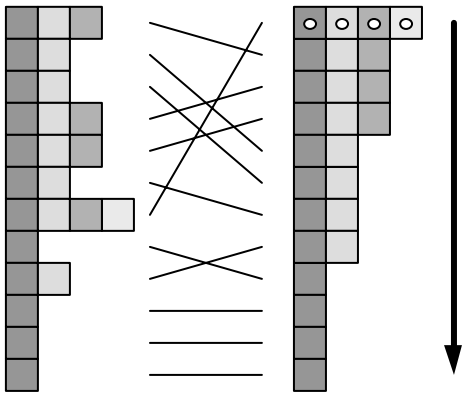
The compressed row storage (CRS) [13] matrix storage format is highly memory-efficient, however the innermost loop is relatively short due to matrix-vector operations as follows :

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

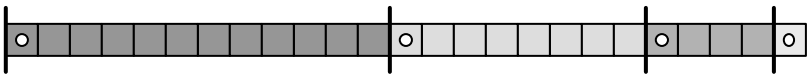
The following loop exchange is then effective for obtaining a sufficiently long innermost loop :

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

Descending-order jagged diagonal storage (DJDS) [12] is suitable for this type of operation and involves permuting rows into an order of decreasing number of non-zeros, as in Fig. 5(a). As elements on the same hyperplane are independent, performing this permutation inside a hyperplane does not affect results. Thus, a 1D array of matrix coefficients with continuous memory access can be obtained, as shown in Fig. 5(b).



(a) Permutation of rows into order of decreasing number of non-zeros

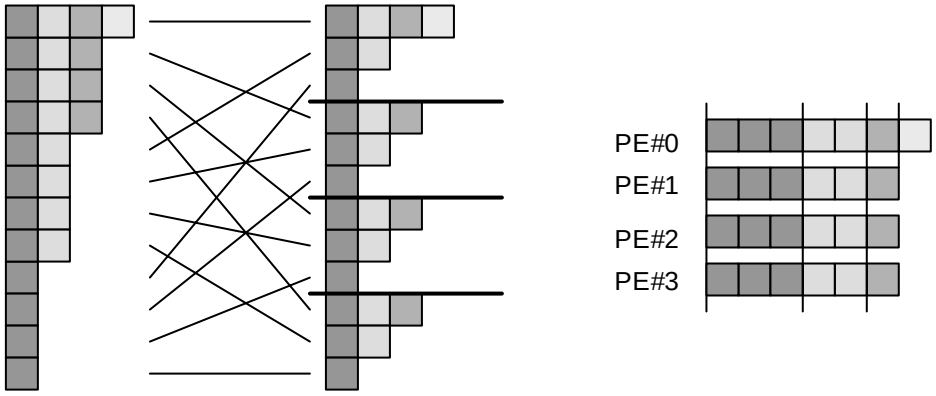


(b) 1D array of matrix coefficient

Fig. 5 DJDS reordering for efficient vector/parallel processing

3.3 Distribution over SMP Nodes : Parallel DJDS Reordering

The 1D array of matrix coefficients with continuous memory access is suitable for both parallel and vector computing. The loops for this type of array are easily distributed to each PE in an SMP node via loop directives. In order to balance the computational load across PEs in the SMP node, the DJDS array should be reordered again in cyclic manner. The procedure for this reordering, called parallel PDJDS (PDJDS) is described in Fig. 6 :



(a) (b)

Fig. 6 PDJDS reordering for load-balanced parallel processing in an SMP node : Example with 4 PEs per SMP node (a) Cyclic reordering (b) 1D array assigned to each PE after reordering and load-balancing.

3.4 Summary of Reordering Methods

The reordering procedures for increasing parallel/vector performance of the SMP cluster architecture described in this section are summarized as follows :

(1) RCM reordering on the original local matrix for independent sets.
(2) CM reordering to obtain loops whose length is sufficiently long and uniform.
(3) DJDS reordering for efficient vector processing, producing 1D arrays of coefficients with continuous memory access.
(4) Cyclic reordering for load-balancing among PEs on an SMP node.
(5) PDJDS/CM-RCM reordering is complete.

The typical loop structure of the matrix-vector operations for PDJDS /CM-RCM reordered matrices based on the pseudo-vector and parallel directives of the Hitachi SR8000 is described in the following :

```
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

The Hitachi SR8000 is a distributed-memory parallel system with 4 to 128 configurable nodes. The nodes are connected by a high-speed multidimensional crossbar network and 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 at the Computing Center of the University of Tokyo was employed. Each node provides 8 GFLOPS peak performance, and the total peak performance is approximately 1 TFLOPS[10].

4.1 Cooperative microprocessors (COMPAS)[9]

This functionality provides high-speed simultaneous activation of multiple processors in a node. 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, allowing the user to code data without being aware of hardware architecture. Parallelization of vector operations simplifies conversion from the standard vector operations.

4.2 Pseudo-Vectorization[9]

High-speed numerical computations in the microprocessor are achieved by pseudo-vectorization. Each microprocessor in a node pipelines data from memory without interrupting subsequent instructions. Therefore, high-speed large-scale computing is possible by supplying a large amount of data to the computing element from memory.

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

As a solution to this performance reduction, 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 in advance in a floating-point register, and loading is completed while the loop that references the data is performing calculations from previous iterations. (preload optimizing)
- The data is transferred is advance into a memory cache, and the transfer is completed while the loop that references the data is performing calculations from previous iterations. (prefetch optimizing)

5. Examples

The proposed methods were applied to large-scale 3D solid mechanics example cases, as described in Fig. 7, which represent linear elastic problems with homogeneous material property and boundary conditions. Each element is a cube with unit edge length, and each node has 3 DOF. therefore there are 3×Nx×Ny×Nz DOF in total for the problem.

For this problem, 3×3 Block ICCG(0) with PDJDS/MC-RCM reordering is applied with full LU factorization for each 3×3 diagonal block. One ASDD operation is applied to each iteration.

Vector performance was evaluated on an NEC SX-4 (JAERI/CCSE) and a Hitachi SR2201 (University of Tokyo), and SMP parallel performance was tested on the Hitachi SR8000. In each

case, the number of colors was set to 99, corresponding to an average vector length of (total number of FEM nodes) / (number of PEs or SMP nodes × 99 × NPE) where NPE = 1 for SX-4 and SR2201, and NPE=8 for SR8000.

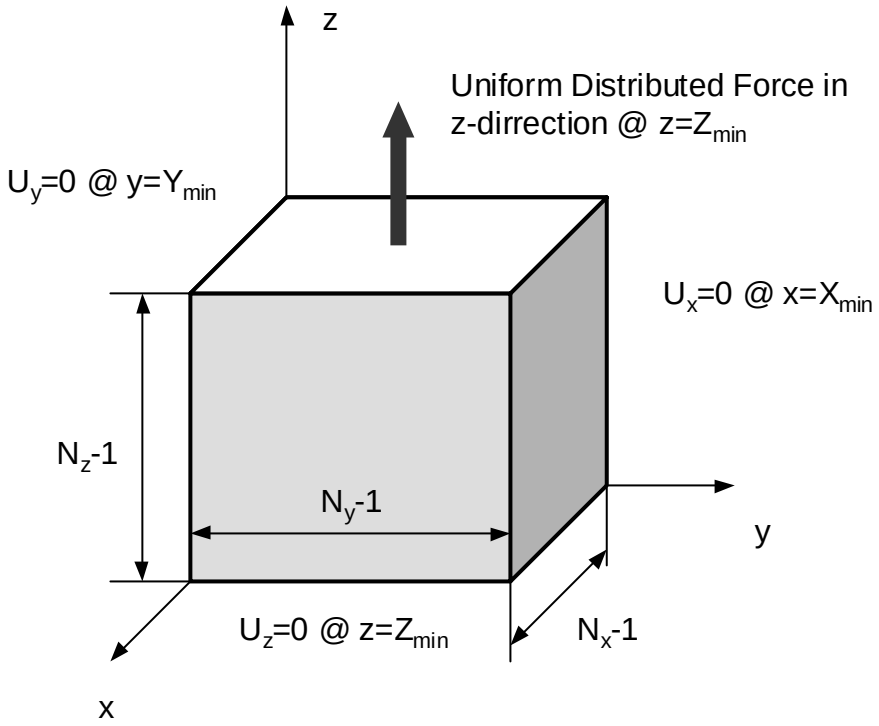


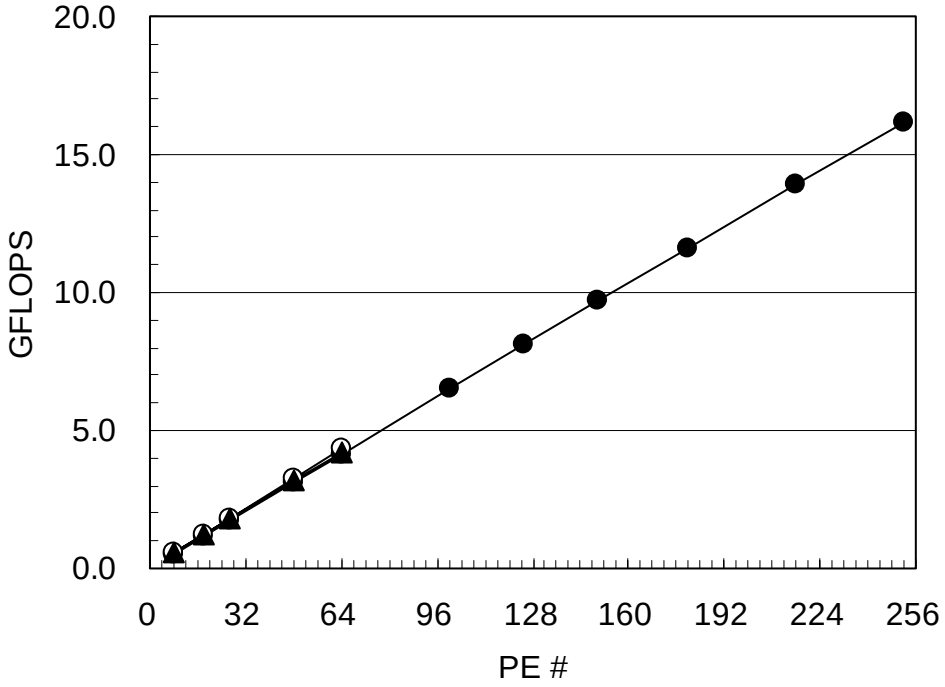
Fig. 7 Problem definition and boundary conditions for 3D solid mechanics example cases. Linear elastic problem with homogeneous material property and boundary conditions. Each element is cube with unit edge length. Problem has 3×Nx×Ny×Nz DOF in total.

5.1 Vector/Vector Parallel Performance

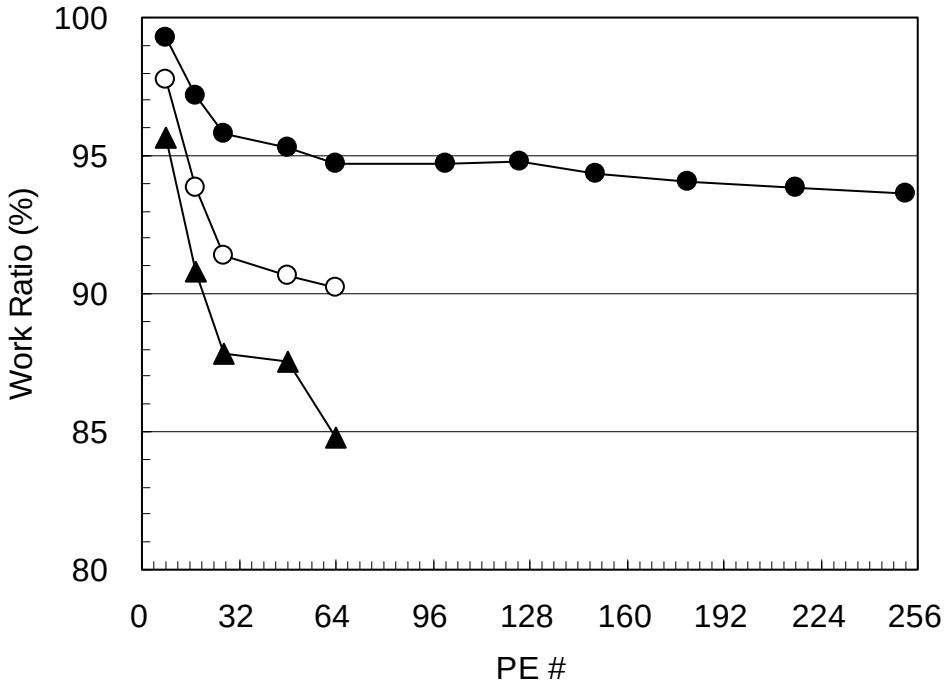
Before computation on the Hitachi SR8000, vector performance was evaluated on an NEC SX-4 and Hitachi SR2201. Parallelization for SMP nodes was not applied ; matrices are therefore reordered by DJDS/CM-RCM.

An example including 41^3 nodes (206,763 DOF) was solved using 1 processor of the NEC SX-4, achieving 969 MFLOPS performance for the linear solver component with peak performance of 2 GFLOPS (48.5% of peak performance).

Evaluations for parallel computing were conducted on the Hitachi SR2201 at the University of Tokyo[10]. Pseudo vectorization can also be applied on the SR2201, and each PE performs as a vector processor. Figure 8 shows the GFLOPS rate and work ratio (real computation time/elapsed execution time including communication) for various problem sizes. In these computations, the problem size for 1 PE was fixed. The largest case was 27,168,372 DOF on 252 PEs. A performance of 16.2 GFLOPS was achieved. Each processor is capable of 300 MFLOPS peak performance, and the total peak performance of the system was 75.6 GFLOPS with 252 PEs. The 16.2 GFLOPS performance then corresponds to 21.4% of the peak performance. Figure 8(b) shows that the work ratio is higher than 90% if the problem size for 1 PE is sufficiently large, more than 24,000 DOF in this case.



(a) PE# ~ GFLOPS rate relationship



(b) PE# ~ Work ratio relationship

Fig.8 GFLOPS rate and work ratio for various problem sizes on Hitachi SR2201 with DJDS/CM-RCM reordering. Problem size/PE is fixed. Largest case is 27,168,372 DOF on 252 PEs.

●($3\times33^3 = 107{,}811$)DOFs /PE ○($3\times20^3 = 24{,}000$) ▲($3\times15^3 = 10{,}125$)

5.2 SMP Parallel Performance

The following cases were tested on the Hitachi SR8000 :

(1) The increase in speed for fixed problem size (3×128^3 DOF) using between 1 and 16 SMP nodes.
(2) Communication/synchronization overhead for intra-SMP node parallellization for various problem sizes using 1 SMP node.
(3) Effect of matrix storage and reordering for various problem sizes using 1 SMP node.
(4) Performance evaluation for various problem sizes using 1 to 16 SMP nodes

Figure 9 shows the results of (1). In this example, the size of the entire problem was fixed at 3×128^3 (6,291,456) DOF, and the increase in speed was evaluated for 1 to 16 SMP nodes. The number of iterations for convergence ($\varepsilon = 10^{-8}$) was 333 (1-node), 337 (2-nodes), 338 (4-nodes), 341 (8-nodes) and 347 (16-nodes), indicating that the number of iterations remains almost constant as the number of nodes increases. This is due to the ASDD. In this case, there is a superlinear speedup of convergence as the number of processors increases for the 2-, 4-, and 8-node cases. Speedup rate at 16 SMP nodes was 14.2, which corresponds to the 88.8% of the linear (ideal) speedup.

Figure 10 shows the results of (2). Communication/synchronization overhead occurs for parallel processing in each SMP node. The work ratio was measured for various problem sizes from 3×16^3 (12,288) DOF to 3×128^3 (6,291,456) DOF on 1 SMP node. Measurements were made using the XCLOCK system subroutine of the Hitachi compiler[9]. The results show that overhead is more than 30% for the smallest problem size, and less than 10% for the problem size of 3×40^3 (192,000) DOF (24,000 DOF/PE) and less than 5% for 3×64^3 (786,432) DOF (98,304 DOF/PE). According to these results, communication/synchronization overhead for intra-SMP node communication is almost negligible if the problem size is sufficiently large.

Figure 11 shows the results of (3), demonstrating the effect of reordering. In this case, the following 3 cases were compared :

- PDJDS/CM-RCM reordering
- Parallel descending-order compressed row storage (PDCRS) /CM-RCM reordering
- CRS without reordering

PDCRS/CM-RCM reordering is identical to PDJDS/CM-RCM except for the storage of matrices in a CRS manner after permutation of rows into the order of decreasing number of non-zeros, where the length of the innermost loop is shorter than that for PDJDS. The elapsed execution time was measured for various problem sizes from 3×16^3 (12,288) DOF to 3×128^3 (6,291,456) DOF on 1 SMP node. PDCRS is faster than PDJDS for smaller problems, but PDJDS outperforms PDCRS for larger problems as a result of pseudo vectorization. The performance of PDJDS decreases if the problem size is larger than 10^6 DOF, whereas this curve would be flat for vector processors. Therefore, this decrease in performance is due to *pseudo vectorization* and the performance of the compiler in Hitachi SR8001. The cases without reordering exhibit very poor performance. Parallel computation is impossible for forward/backward substitution (FBS) in the IC factorization process even in the simple geometry examined in this study. This FBS process represents about 50% of the total computation time. If this process is not parallelized, the performance reaches only about 20% of that with reordering. The number of iterations for convergence is also larger for cases without reordering, as shown in Table 2.

Figure 12 shows the results of (4). This figure can be compared with Fig. 8 for the Hitachi SR2201. The problem size is fixed for one SMP node and the number of nodes was varied be-

tween 1 and 16. The largest problem size was $16\times3\times128^3$ (100,663,296) DOF, for which the performance was about 20 GFLOPS, corresponding to 15.6% of the total peak performance of the 16 SMP nodes. Figure 12(b) shows that the performance at small problem sizes per SMP node (3×32^3=98,304 DOF), was almost 50% of that for the larger problems.

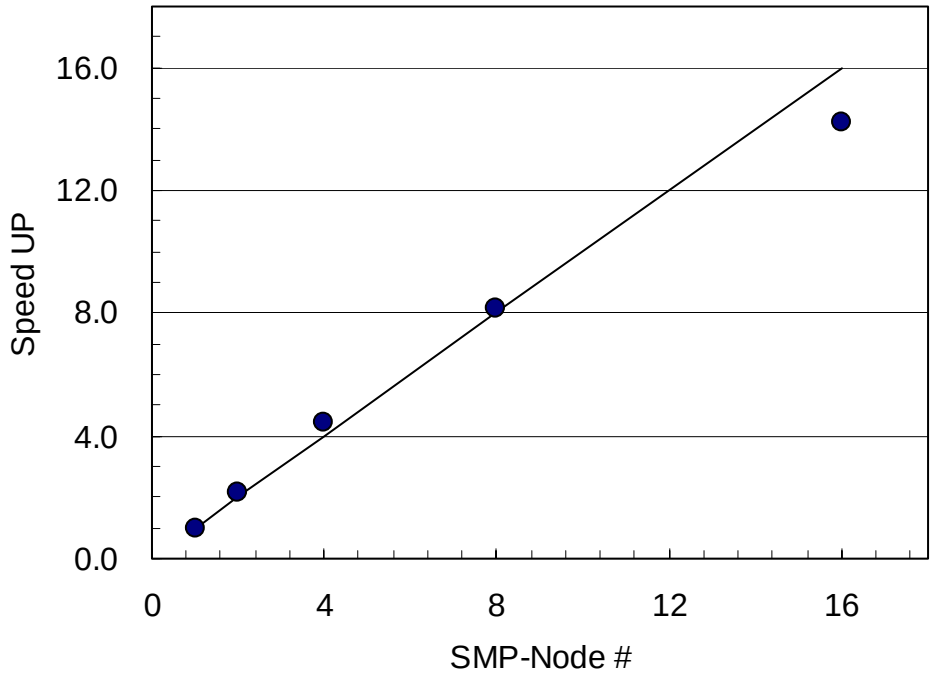


Fig. 9 Relationship between number of SMP nodes and the speedup on the Hitachi SR8000 with DJDS/CM-RCM reordering. The total problem size is fixed at 3×128^3 (6,291,456) DOF. Speedup rate for 16 SMP nodes is 14.2. A super-linear effect is observed for the 2-, 4- and 8-node cases.

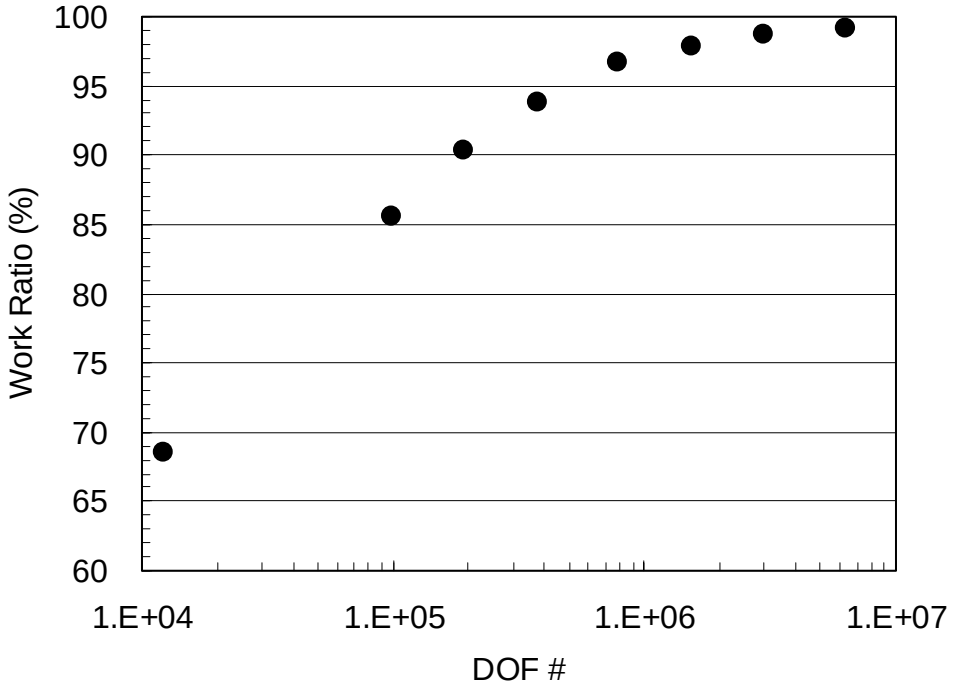


Fig. 10 Work ratio for various problem sizes on the Hitachi SR8000 with 1 SMP node woth PDJDS/CM-RCM reordering. The work ratio is more than 90% if the problem size is 3×40^3 (192,000) DOF (24,000 DOF/PE).

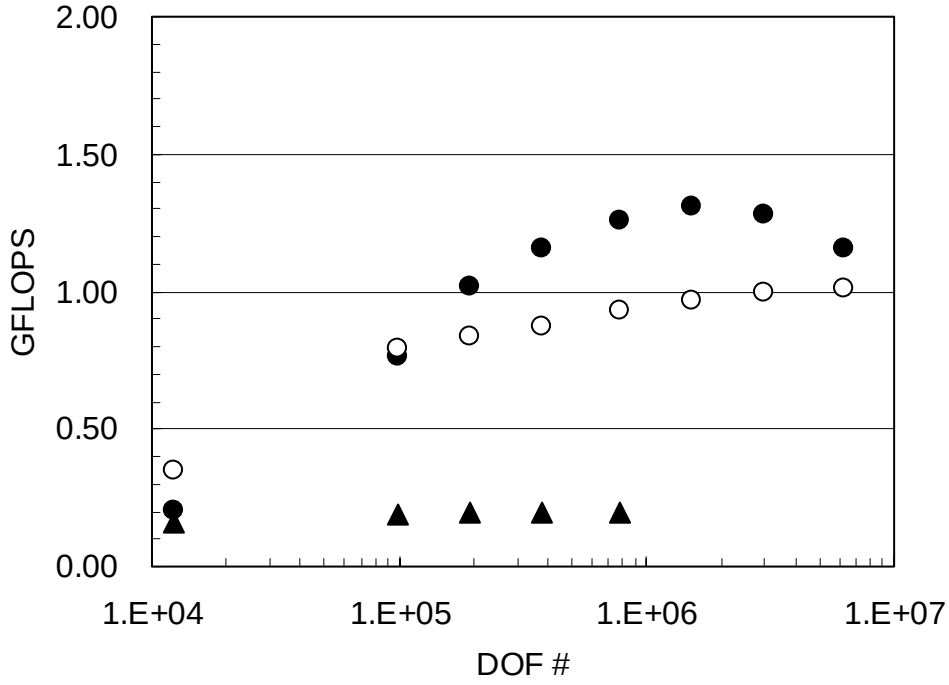
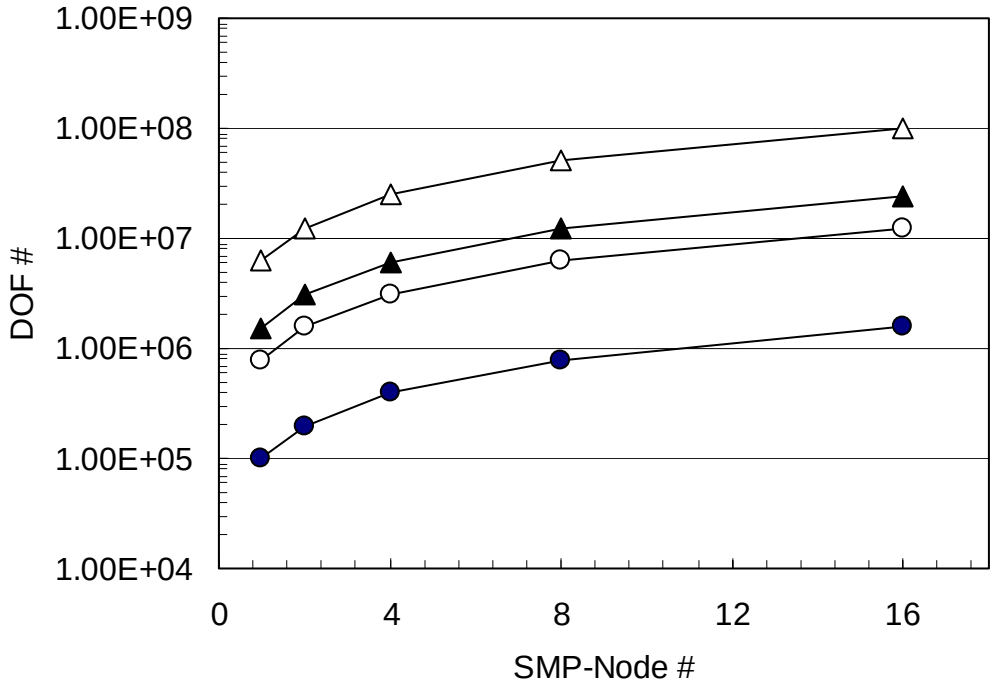


Fig. 11 Effect of coefficient matrix storage method and reordering for various problem sizes on the Hitachi SR8000 with 1 SMP node. The performance of the solver without reordering is very low due to synchronization overhead during forward/backward substitution for the IC factorization. PDCRS/CM-RCM performs better than PDJDS/CM-RCM for small problems, but performs worse for larger problems due to short innermost loops. The performance of PDJDS/CM-RCM decreases at problem sizes larger than 10^6 DOF as a result of pseudo-vector processing of the Hitachi SR8000.

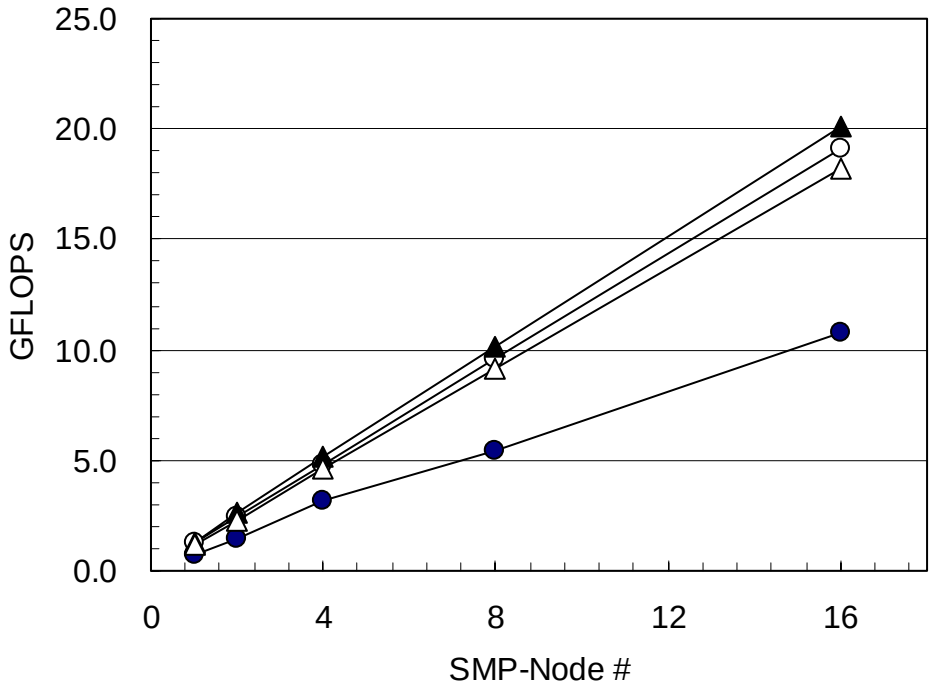
●PDJDS/CM-RCM ○PDCRS/CM-RCM ▲CRS no re-ordering

Table 2. Effect of coefficient matrix storage method and reordering for various problem sizes on the Hitachi SR8000 with 1 SMP node. Number of iterations for convergence $\varepsilon=10^{-8}$

DOF #	With Reordering	Without Reordering
$3x16^3$= 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 for various problem sizes on the Hitachi SR8000. Problem Size/PE is fixed. Largest case is 100,663,296 DOF on 16 SMP nodes (128 PEs). Maximum performance is 20.1 GFLOPS (Peak Performance = 128 GFLOPS)

● ($3\times32^3 = 98{,}304$) DOFs /SMP-Node ○ ($3\times64^3 = 786{,}432$)

▲ ($3\times80^3 = 1{,}536{,}000$) △ ($3\times128^3 = 6{,}291{,}456$)

6. Conclusion

In this study, an efficient parallel iterative method for unstructured grids was developed for SMP cluster architectures on the GeoFEM platform using *loop directives + message passing* type parallel programming model with the following 3 level hierarchy :

- Inter-SMP node MPI
- Intra-SMP node Compiler directives for parallelization
- Individual PE Compiler directives for vectorization/pseudo vectorization

Simple 3D elastic linear problems with more than 10^8 DOF were solved by 3×3 block ICCG(0) with additive Schwarz domain decomposition and PDJDS/CM-RCM reordering on 16 SMP nodes of a Hitachi SR8000, achieving performance of 20 GFLOPS. PDJDS/CM-RCM reordering provides excellent vector and parallel performance in SMP nodes. Without reordering, parallel processing of forward/backward substitution in IC/ILU factorization was impossible due to global data dependency even in the simple examples in this study. Communication/synchronization overhead in a SMP node is less than 10% if the problem size is 3×40^3 (192,000) DOF (24,000 DOF/PE) or larger.

The proposed method was also tested on an NEC SX-4, achieving performance of 969 MFLOPS (48.5% of peak performance) for a problem with 2×10^5 DOF using a single processor. Vector/parallel efficiency was evaluated on a Hitachi SR2201 with pseudo vectorization for various problem sizes. The largest case was 2.72×10^7 DOF on 252 PEs at 16.2 GFLOPS (21.4% of peak performance). The work ratio for parallel computing was found to be higher than 90% if the problem size for 1 PE was sufficiently large, more than 24,000 DOF in this case.

Additive Schwarz domain decomposition was applied to GeoFEM's parallel iterative solvers with localized preconditioning, providing robustness with respect to the localized preconditioning. The number of iterations for convergence remains constant even if the number of partition increases for fixed-size problems.

In this study, an effective hybrid parallel programming model for SMP cluster architecture was developed, yet high computational performance was not realized (16% of peak performance). Therefore, the authors intend to further optimize the methods, particularly with respect to single PE performance. Future study will also include :

- porting the methods to other SMP cluster hardware, and
- applying the methods to real-world problems with more complicated geometries.

Acknowledgments

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

Furthermore the authors would like to thank Professor Yasumasa Kanada (Computing Center, The University of Tokyo) for 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] Accelerated Strategic Computing Initiative (ASCI) Web Site : http://www.llnl.gov/asci/
[2] Earth Simulator Research and Development Center Web Site : http://www.es.jamstec.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.