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

Kengo Nakajima[1] and Hiroshi Okuda[2]

[1] Research Organization for Information Science and Technology (RIST)
2-2-54 Naka-Meguro, Meguro-ku, Tokyo 153-0061, Japan
nakajima@tokyo.rist.or.jp
[2] Department of Quantum Engineering and System Sciences
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
okuda@garlic.q.t.u-tokyo.ac.jp

Abstract. An efficient parallel iterative method for unstructured grids developed by the authors for 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 by OpenMP for intra-SMP node parallelization and vectorization for each processing element (PE). Simple 3D elastic linear problems with more than 8×10^8 DOF have been solved by 3×3 block ICCG(0) with additive Schwarz domain decomposition and PDJDS/CM-RCM reordering on 128 SMP nodes of Hitachi SR8000/MPP parallel computer, achieving performance of 335.2 GFLOPS. The PDJDS/CM-RCM reordering method provides excellent vector and parallel performance in SMP nodes.

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 [5]. A significant amount of research on this issue has been conducted in recent 2 or 3 years [6][7], but most studies

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 Hitachi SR8000 [9] parallel computers 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 OpenMP for parallelization
- Individual PE Compiler directives for (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 and reordering techniques for parallel and vector computation on SMP nodes, and present the results for an application to 3D solid mechanics.

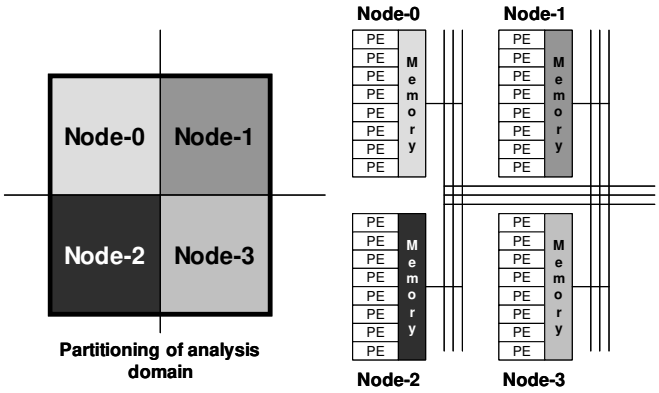


Fig. 1. Parallel FEM computation on SMP cluster architecture. Each partition corresponds to an SMP node.

2 Parallel Iterative Solvers in GeoFEM

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

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:

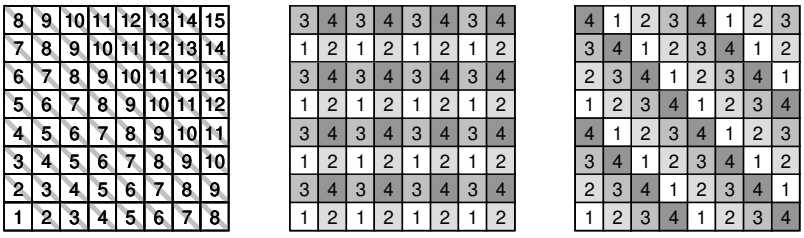
- Local operations and no global dependency
- Continuous memory access
- Sufficiently long loops

For unstructured grids, in which data and memory access patterns are very irregular, the reordering technique is very effective for achieving highly 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 hyper-

plane (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. 2(a)) reordering provides fast convergence of IC/ILU-preconditioned Krylov iterative solvers, yet with irregular hyperplane size. For example in Fig. 2(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. 2(b)). However, it is widely known that the convergence of IC/ILU-preconditioned Krylov iterative solvers with multicolor ordering 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 2(c) shows an example of CM-RCM reordering. In this case, there are 4 colors ; the 1st, 5th, 9th and 13th hyperplanes in Fig. 2(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. 2 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:

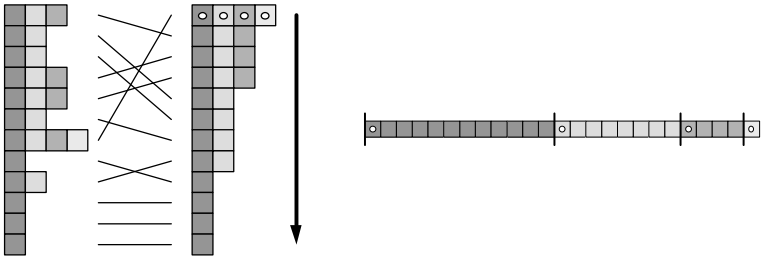
```
do i= 1, N
  do j= 1, NU(i)
    (operations)
    F(i)= F(i) + A(k1)*X(k2)
  enddo
enddo
```

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. 3(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. 3(b).



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

(b) 1D array of matrix coefficient

Fig. 3 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 DJDS (PDJDS) is described in Fig. 4:

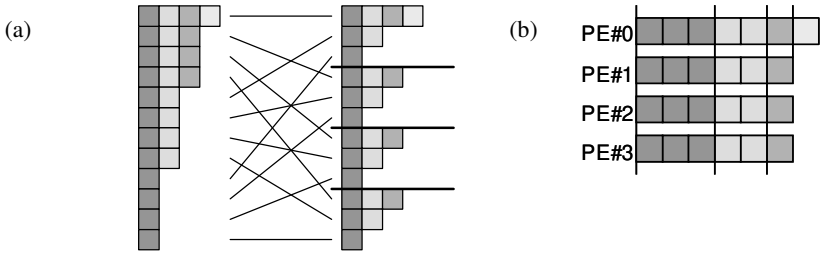


Fig. 4 PDJDS reordering for an SMP node : Example with 4 PEs per SMP node (a) Cyclic reordering (b) 1D array assigned to each PE after reordering and load-balancing.

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 OpenMP directives of the Hitachi SR8000 is described in the following:

```
do col= 1, COLORtot
  do j= 1, NUmax(col)
!$OMP PARALLEL DO PRIVATE (iS, iE, i) : OpenMP Directive
    do pe= 1, SMP_PE_tot
      iS= NstartU(col,j,pe)
      iE= NendU  (col,j,pe)
*VOPTION, INDEP  : Directive for Vectorization
      do i= iS, iE
        (operations)
      enddo
    enddo
  enddo
enddo
```

4 Examples

The proposed methods were applied to large-scale 3D solid mechanics example cases, as described in Fig. 5, 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.

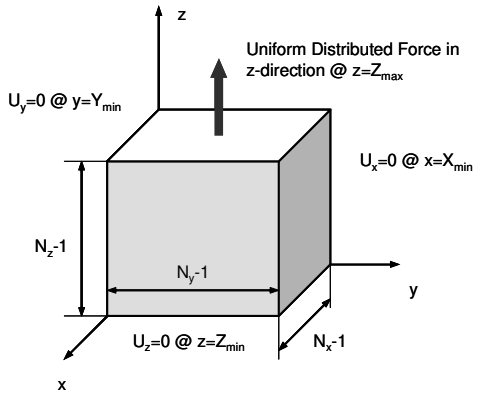


Fig. 5 Problem definition and boundary conditions for 3D solid mechanics example cases.

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(number of PEs on each SMP node)) where NPE = 1 for SX-4 and SR2201, and NPE=8 for SR8000.

4.1 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; ma-

trices 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) [3].

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. The largest case was 196,608,000 DOF on 1024 PEs. A performance of 68.7 GFLOPS was achieved. Total peak performance of the system was 300 GFLOPS with 1024 PEs [3]. The 68.7 GFLOPS performance then corresponds to 22.9% of the peak performance. 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 [3].

4.2 SMP Parallel Performance

The following cases were tested on the Hitachi SR8000 at the University of Tokyo. 2 systems were used. One is Hitachi SR8000/128 (8 PEs, 8GFLOPS peak performance and 8GB memory for each node. 128 nodes (1024 PEs), 1.0TFLOPS peak performance and 1.0TB memory for total system). The other is Hitachi SR8000/MPP (8PEs, 14.4GFLOPS peak performance and 16GB memory for each node. 128 nodes (1024 PEs), 1.8TFLOPS peak performance and 2.0TB memory for total system):

(1) The increase in speed for fixed problem size (3×128^3=6,291,456 DOF) using between 1 and 8 SMP nodes (SR8000/MPP).
(2) Communication/synchronization overhead for intra-SMP node parallelization for various problem sizes using 1 SMP node (SR8000/MPP).
(3) Effect of matrix storage and reordering for various problem sizes using 1 SMP node (SR8000/128).
(4) Performance evaluation for various problem sizes using 1 to 128 SMP nodes (SR8000/MPP).

Figure 6 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 8 SMP nodes. The number of iterations for convergence ($\varepsilon= 10^{-8}$) was 333 (1-node), 337 (2-nodes), 338 (4-nodes), and 341 (8-nodes), indicating that the number of iterations remains almost constant as the number of nodes increases. This is due to the ASDD. Speedup rate at 8 SMP nodes was 6.49, which corresponds to the 81.1% of the linear (ideal) speedup. Speedup effect for many nodes is worse than the ideal speedup due to smaller problem size per node.

Figure 7 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 8 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 effect of vector length is much more significant than that of comm./sync. overhead in SMP node shown in Fig.7. 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 less than 20% of that with reordering. The number of iterations for convergence is also larger for cases without reordering.

Figure 9 and 10 show the results of (4). The problem size is fixed for one SMP node and the number of nodes was varied between 1 and 128. The largest problem size was $128\times3\times128^3$ (805,306,368) DOF, for which the performance was about 335.2 GFLOPS, corresponding to 18.6% of the total peak performance of the 128 SMP nodes. Figure 9 and 10 show 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. Figure 11 shows the results with various problem sizes on 128 SMP nodes. Parallel work ratio among SMP nodes by MPI is more than 95% if problem size is sufficiently large.

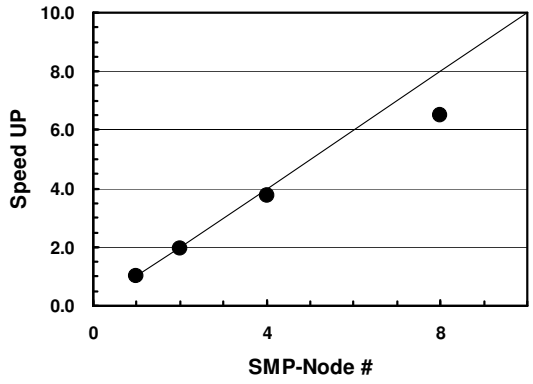


Fig. 6 Relationship between number of SMP nodes and the speedup on the Hitachi SR8000/MPP with DJDS/CM-RCM reordering. The total problem size is fixed at 3×128^3 (6,291,456) DOF. Speedup rate for 8 SMP nodes is 6.49.

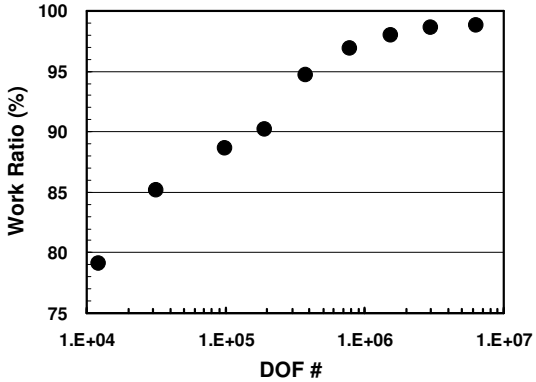


Fig. 7 Work ratio for various problem sizes on the Hitachi SR8000/MPP with 1 SMP node woth PDJDS/CM-RCM reordering. The work ratio is more than 90% if the problem size is 3×40^3 (192,000) DOF (24,000 DOF/PE).

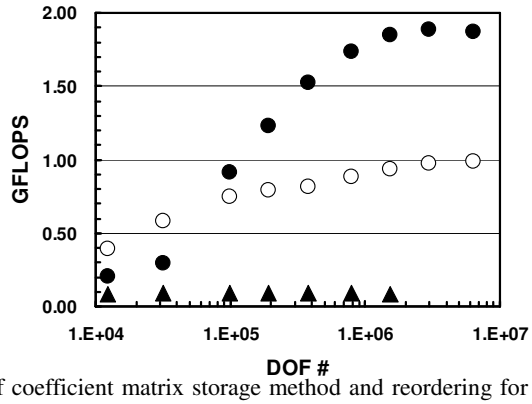


Fig. 8 Effect of coefficient matrix storage method and reordering for various problem sizes on the Hitachi SR8000/128 with 1 SMP node. The performance of the solver without reordering is very low due to synchronization overhead during forward/backward substitution for the IC factorization (BLACK Circles: PDJDS/CM-RCM, WHITE Circles: PDCRS/CM-RCM, BLACK Triangles: CRS no reordering).

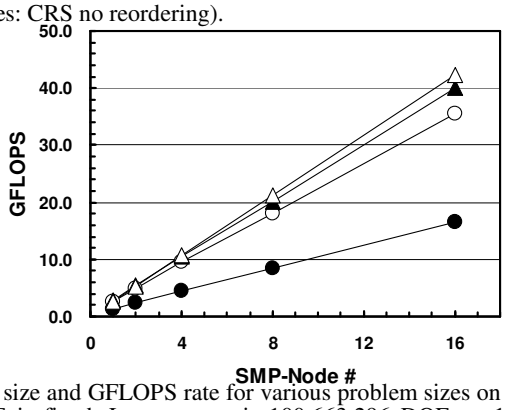


Fig. 9 Problem size and GFLOPS rate for various problem sizes on the Hitachi SR8000/MPP. Problem Size/PE is fixed. Largest case is 100,663,296 DOF on 16 SMP nodes (128 PEs). Maximum performance is 42.4 GFLOPS (Peak Performance = 230.4 GFLOPS). (BLACK Circles: 3×32^3=98,304 DOF/SMP node, WHITE Circles: 3×64^3, BLACK Triangles: 3×80^3, WHITE Triangles: 3×128^3).

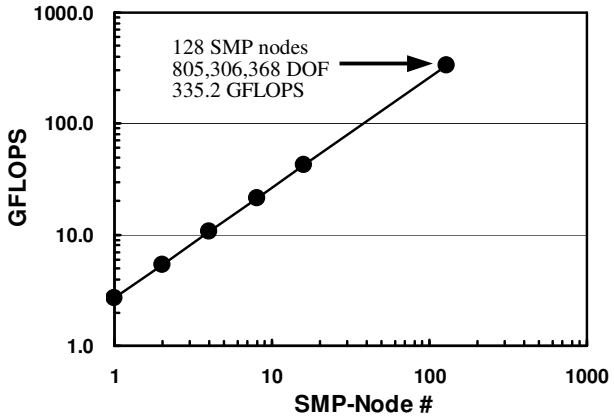


Fig. 10 Problem size and GFLOPS rate for various problem sizes on the Hitachi SR8000/MPP. Problem Size/PE is fixed as 6,291,456 DOF. Largest case is 805,306,368 DOF on 128 SMP nodes (1024 PEs). Maximum performance is 335.2 GFLOPS (Peak Performance = 1.8 TFLOPS)

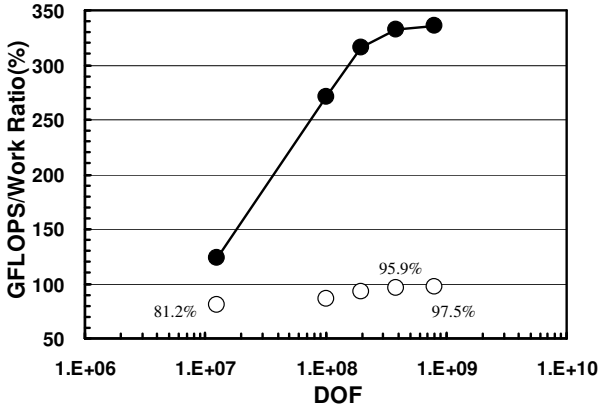


Fig. 11 Problem size and GFLOPS rate/iterations for convergence ($\varepsilon=10^{-8}$) in various problem sizes on 128 SMP nodes of the Hitachi SR8000/MPP. The largest case is 805,306,368 DOF on 128 SMP nodes (1024 PEs). Maximum performance was 335.2 GFLOPS (Peak Performance = 1.8 TFLOPS). (BLACK Circles: GFLOPS rate, WHITE Circles: Parallel work ratio (inter SMP nodes)). Parallel work ratio is more than 95 % if problem size is sufficiently large.

5 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 OpenMP for parallelization
- Individual PE Compiler directives for (pseudo) vectorization

Simple 3D elastic linear problems with more than 8×10^8 DOF were solved by 3×3 block ICCG(0) with additive Schwarz domain decomposition and PDJDS/CM-RCM reordering on 128 SMP nodes of a Hitachi SR8000/MPP, achieving performance of 335.2 GFLOPS (18.6% of peak performance). 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 parallel work ratio for inter SMP node communication was found to be higher than 95% if the problem size for each node was sufficiently large. 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 and on a Hitachi SR2201 with 68.7 GFLOPS (22.9% of peak performance) for a problem with 1.97×10^8 DOF using 1024 processors.

In this study, an effective hybrid parallel programming model for SMP cluster architecture was developed. Future study includes:

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

Acknowledgements

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 (Information Technology 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.

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. Information Technology 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., Bjφrstad, 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.