doi: 10.32131/jes.2.41

Journal of the Earth Simulator, Volume 2, March 2005, 41–54

Seismic Response Analysis of Nuclear Pressure Vessel Model with ADVENTRUE System on the Earth Simulator

Masao Ogino[1]*, Ryuji Shioya[1], Hiroshi Kawai[2] and Shinobu Yoshimura[3]

[1] *Faculty of Engineering, Kyushu University, 6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-8651, Japan*
[2] *Faculty of Science and Technology , Keio University, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama-shi, Kanagawa-ken 223-8522, Japan*
[3] *Graduate School of Frontier Sciences, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan*

(Received November 8, 2004; Revised manuscript accepted January 20, 2005)

Abstract We have been developing an advanced general-purpose computational mechanics system, named ADVENTURE, which is designed to be able to analyze a three dimensional finite element model of arbitrary shape over 100 million Degrees Of Freedom (DOF) mesh. The one of main process modules for solid analysis, named ADVENTURE_Solid, is based the hierarchical domain decomposition parallel algorithm and employs the balancing domain decomposition as a solution technique for linearized equations. The ADVENTURE_Solid has been successfully implemented on a single PC, PC clusters and MPPs with high parallel performances. In order to realize a virtual demonstration test, the solid module is implemented on the Earth Simulator with minor modification for a vector processor and applied for an implicit dynamic elastic analysis such as seismic response of a precise nuclear pressure vessel model of 35 million DOF mesh with 128 nodes (1,024 Processing Elements), and succeeded in solving 300 unsteady steps in about 4.3 hours. Furthermore, the system is applied for a static elastic analysis of a nuclear pressure vessel model of 100 million DOF mesh in about 8.5 minutes on the 256 nodes (2,048 PEs), and archived 5.08 TFLOPS, which is 31.75% of the peak performance.

and succeeded in solving 300 unsteady steps in about 4.3 hours. Furthermore, the system is applied for a static elastic analysis of a nuclear pressure vessel model of 100 million DOF mesh in about 8.5 minutes on the 256 nodes (2,048 PEs), and archived 5.08 TFLOPS, which is 31.75% of the peak performance.

Keywords: parallel finite element method, balancing domain decomposition, seismic response analysis, ADVENTRUE system

1. Introduction

Various general-purpose computational mechanics systems have been developed in the last three decades to quantitatively evaluate mechanical / physical phenomena such as deformation of solid, heat transfer, fluid flow and electromagnetics. Nowadays such systems are regarded as infrastructural tools for the present industrialized society. The existing systems, however, cannot be used with Massively Parallel Processors (MPPs) with the order of 100–10,000 Processing Elements (PEs), which are dominating the high-performance computing market in the 21st century, as they were developed for single-processor computers, which took leadership an age ago. Neither can the current systems be used in heterogeneous parallel and distributed computing environments. Owing to the fact, they can deal with only medium scale problems with millions Degrees Of Freedom (DOF) at most.

The ADVENTURE project [1, 2] is one of the research projects in the "Computational Science and Engineering" field selected for the "Research for the Future (RFTF)" Program sponsored by the Japan Society for the Promotion of Science (JSPS) [3, 4]. In the project we have been developing an advanced general-purpose computational mechanics system named ADVENTURE since August 1997. The system is designed to be able to analyze a three-dimensional (3D) finite element model of arbitrary shape over 100 million DOF mesh, and additionally to enable parametric and non-parametric shape optimization [2]. The first version of the ADVENTURE system has been released from the project website as open source software since March 2002 [1]. About 1,600 registered users in academia and industries are now test-

* ***Corresponding author***: *Dr. Masao Ogino, Faculty of Engineering, Kyushu University, 6–10–1 Hakozaki, Higashi-ku, Fukuoka 812-8651, Japan. E-mail: ogino@mech.kyushu-u.ac.jp*

ing the programs, while one private company has developed and released its commercial version named ADVentureCluster [5].

Domain decomposition based parallel algorithms are implemented in pre-processes (domain decomposition), main processes (system matrix assembling and solutions) and post-process (visualization), respectively. Especially the Hierarchical Domain Decomposition Method (HDDM) with a preconditioned iterative solver [6–8] is adopted in two of the main modules for solid analysis and thermal conduction analysis, named ADVENTURE_Solid and ADVENTURE_Thermal. The employed preconditioner is the Balancing Domain Decomposition (BDD) type method [9–14]. To efficiently solve a coarse space problem derived from equilibrium conditions for singular problems associated with a number of subdomains appearing in the BDD formulation, a parallel direct solver is employed. The ADVENTURE_Solid has been successfully implemented on a single PC, PC clusters and MPPs such as Hitachi SR8000/MPP [2, 7, 8, 15].

In this paper, the solid analysis module is implemented with minor modification on the Earth Simulator, and applied for an implicit dynamic elastic analysis such as seismic response of a precise nuclear vessel model of 35 million DOF mesh on 128 nodes (1,024 vector-type PEs) and succeeded in solving 300 unsteady steps in 4.3 hours. Furthermore, the system is applied for a static elastic analysis of a nuclear pressure vessel model of 100 million DOF mesh in 8.5 minutes on the 256 nodes (2,048 PEs), and archived 5.08 TFLOPS, which is 31.75% of the peak performance.

2. Overview of ADVENTURE System

The ADVENTURE system consists of pre-, main- and post-processing modules and designs modules that can be used in various kinds of parallel and distributed environments. The system employs the HDDM based massively parallel algorithm as one of the major solution algorithms in order to handle a huge-scale finite element model over 10–100 million DOF efficiently. The system employs module-based architecture and consists of 19 modules. Fig. 1 illustrates the architecture of the ADVENTURE system. The pre-process modules include the surface patch generator named ADVENTURE_TriPatch which converts geometry model data into a collection of triangular surface patch data, a tetrahedral mesh generator [16] named ADVENTURE_TetMesh, an attachment tool of boundary conditions and material properties onto the mesh named ADVENTURE_BCtool, and a domain decomposer of a finite element model named ADVENTURE_Metis. The kernels of the ADVENTURE_Metis are a graph partitioning tool METIS and its parallel version ParMETIS developed in the University of Minnesota [17, 18]. The main process modules, i.e. solvers include an implicit elastic-plastic analysis module named ADVENTURE_Solid [7, 8, 15] which enables large-deformation and implicit dynamic analyses, a thermal conductive analysis module named ADVENTURE_Thermal [14], a thermal-fluid analysis module named ADVENTURE_Fluid, a magnetic analysis module named ADVENTURE_Magnetic [19], an explicit contact-impact analysis module named ADVENTURE_Impact, and a rigid plastic analysis module named ADVENTURE_Forge. The post process module named ADVENTURE_Visual is for parallel visualization of analysis results [20]. Common functions related to finite elements are programmed as class libraries named libFEM. Among the modules, the ADVENTURE_Solid has been improved to apply to MPPs with over thousands of PEs and succeeded in solving a pressure vessel model with 100 million DOF mesh on Hitachi SR8000/MPP [7]. Parallel algorithms implemented in the ADVENTURE_Solid will be explained in detail in the following section.

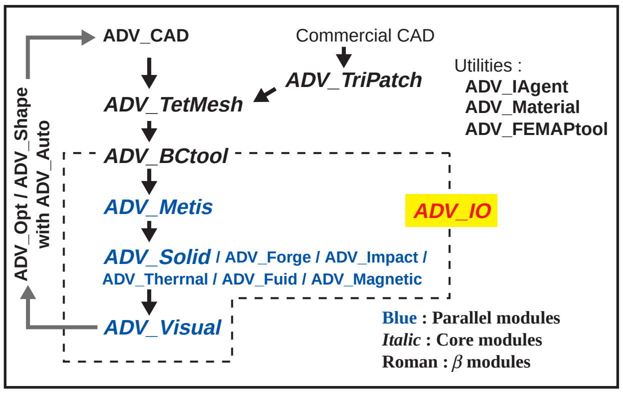

Fig. 1 The module-based architecture of the ADVENTURE system

3. Domain Decomposition Method (DDM)

3.1 Parallel Algorithms Implemented in ADVENTURE_Solid

One of the key technologies implemented in the ADVENTURE_Solid is the HDDM, which enables parallel finite element calculations on various kinds of computing environments [6–8]. Basically in the HDDM, force equivalence and continuity conditions among subdomains are satisfied through iterative calculations such as the Conjugate Gradient (CG) method. Therefore it is indispensable to reduce the number of iterations by adopting some appropriate preconditioner especially when solving large-scale problems with a 10–100 million DOF mesh. The Neumann-Neumann algorithm (N-N) [9] is known as efficient domain decomposition preconditioner for unstructured subdomains. However, its convergence deteriorates with the increasing number of subdomains due to lack of a coarse space problem that takes care of global propagation of error.

The BDD based N-N algorithm proposed by Mandel [10] shows that the equilibrium conditions for singular problems on subdomains result in simple and natural construction of a coarse space problem and that its construction is purely algebraic. The BDD has been applied to solve various phenomena [11, 12]. There are also several researches on parallelism of the BDD and also the Finite Element Tearing and Interconnecting (FETI) [21–26]. However, most problems solved there are still medium scale ones such as sub-millions to a million DOF, and their shape is rather simple. As the DOF of the coarse space problem is directly related to the number of subdomains, it is indispensable to consider the parallelism of the solution process of the coarse space problem as well when solving large-scale problems. The Salinas system [27], which employed FETI-DP method [26], is succeeded in solving large-scale problems such as over 100 million DOF mesh of optical shutter model [28], and shows good performances. However, it is difficult to apply for various kinds of parallel computers because it requires the number of processors in proportional to the degrees of freedom of a analysis model. For more real complex problems, such a technique even with irregular decomposition of domains is indispensable. To overcome these issues, the BDD with a coarse grid correction based a parallel direct method and HDDM is adopted in the ADVENTURE system.

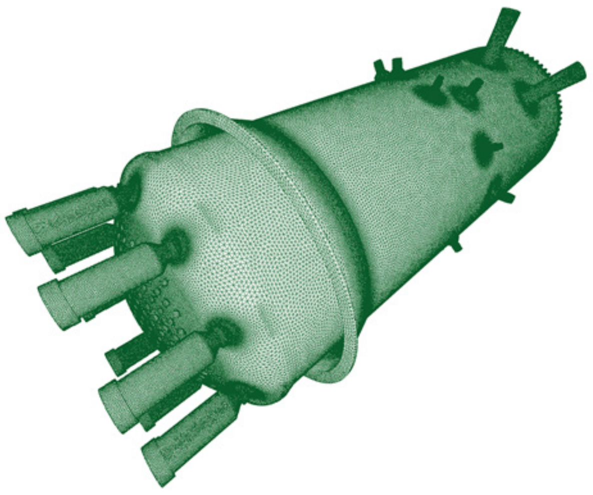
Fig. 2 ABWR model of 35 million DOF mesh

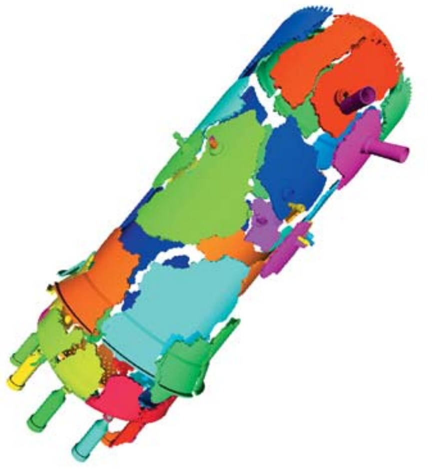
Fig. 3 Part decomposition of ABWR model

3.2 Hierarchical Domain Decomposition Method (HDDM)

In Domain Decomposition Methods (DDM), an analysis model, i.e. a finite element mesh with boundary conditions and material properties, is subdivided into a number of subdomains. The HDDM employs a hierarchical technique to implement the DDM on various parallel computers. In the HDDM, a group of processing elements (PEs) is classified into the following three groups: one Grand Parent PE (Grand), several Parent PEs (Parent or Parents), and many Child PEs (Child or Children). At the same time, the analysis model is first subdivided into several 'parts' whose number is the same as the number of Parents. Then, each part is further subdivided into a number of sub-domains, the number of which can be much larger than that of Children. Fig. 2 shows a 35 million DOF mesh for an Advanced Boiling Water Reactor (ABWR) model, generated by the ADVENTURE_TriPatch and the ADVENTURE_TetMesh. Fig. 3 illustrates an example of the hierarchically decomposed mesh generated by the ADVENTURE_Metis. In this figure, only 128 parts decomposition is shown and each part is divided into subdomains. Owing to the HDDM algorithm, large-scale analysis data can be easily handled by increasing the number of Parents. The main roles of the three kinds of PEs are summarized as follows. The Grand manages all PEs such as synchronization and calculation of the sum of vectors spread over a number of Children. Each Parent stores mesh data and material properties of subdomains, sends / receives subdomain data to / from Child, and iterates loops of the CG method. Each Child performs finite element calculations of the subdomains received from Parent, and sends analyzed data back to the Parent. As the Grand is less work, the role of the Grand is assigned to one of Parents in the ADVENTURE_Solid. Fig. 4 shows the schematic data flow among processors.

According to the design concept of the HDDM, most computation is assigned to Children, while most communication occurs in between Parents and Children. Varying the number of Parents and Children for different kinds of parallel computers, the present HDDM based system can

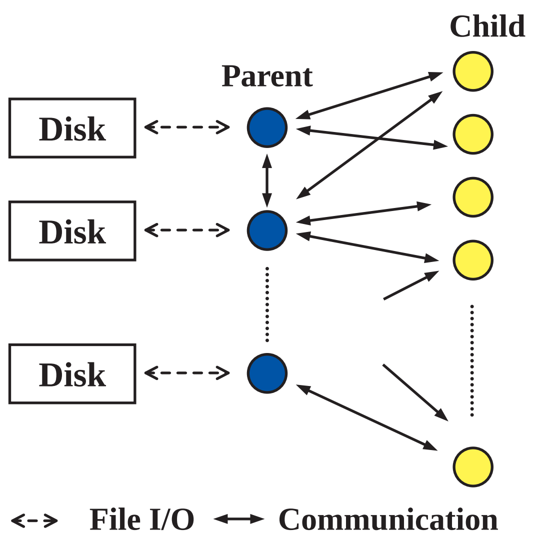

Fig. 4 Schematic data flow in h-mode of the HDDM

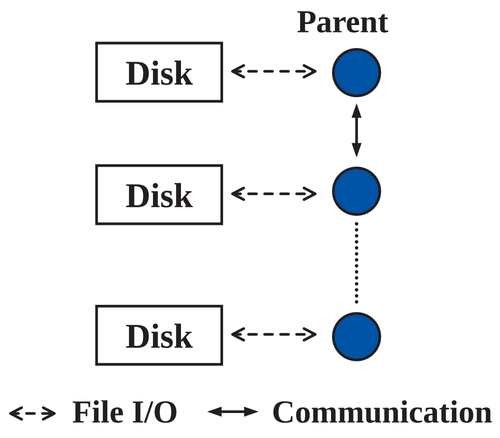

Fig. 5 Schematic data flow in p-mode of the HDDM

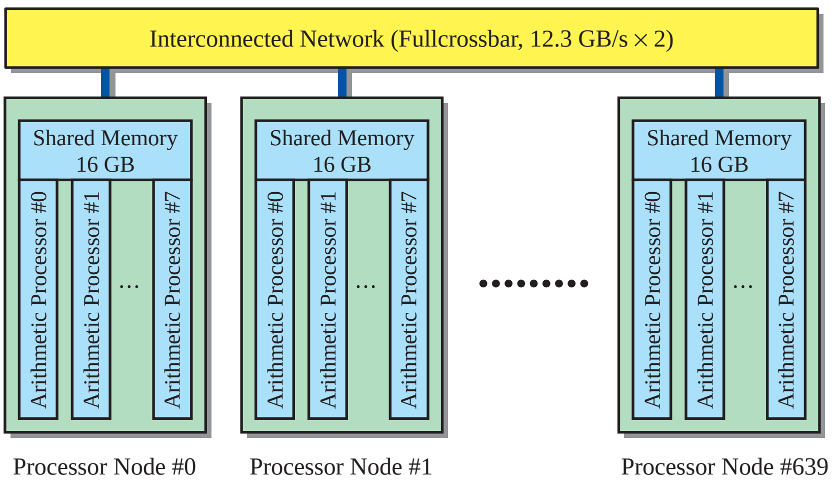

Fig. 6 System configuration of the Earth Simulator

easily achieves high performance. In the HDDM architecture, thanks to the dynamic load balancing technique among Child processors, high parallel performance can be achieved even in heterogeneous computer environments [6–8]. However in this mode, an amount of data communication between Child and Parent tends to be large. To reduce such data communication among Children and Parents, it is useful to assign all tasks of Children and Grand to Parents as well. In this case, load balance becomes static. This analysis mode shown in Fig. 5 is called parallel processors mode (p-mode), while the original analysis mode as shown in Fig. 4 is named hierarchical processors mode (h-mode).

The h-mode keeps load balance dynamically with more communications, and the p-mode, on the other hand, requires few communications with static load balance. Therefore, we can choose the better mode depending on the computer environments. In the case of applying to MPPs like the Earth Simulator, during the calculation, all processor has the same ability, it means, the static load balance is working well.

The Earth Simulator consists of 640 processing nodes (PNs) connected by 640 × 640 single–stage crossbar switches and each PN is a Symmetric Multi Processor (SMP) system with 8 vector type arithmetic processors (APs). The system configuration of the Earth Simulator is illustrated in Fig. 6. According to a hierarchical grouping of PEs, h-mode seems to be suitable to the Earth Simulator. For getting high performance of parallel efficiency, it must be programmed by the "hybrid" parallelization. In the hybrid parallelization, the message communications between PNs is expressed by MPI, and that between APs by microtasking or OpenMP. The ADVENTURE system is aimed to be a general-purpose parallelized CAE system running on various parallel computing environments. Microtasking has dependence on computer environments, and OpenMP is not so popular compared with MPI. Considering to maintain the portability of the ADVENTURE system for various parallel computers, the h-mode and p-mode in the ADVENTURE_Solid are programmed by the "flat" parallelization. In the flat parallelization, the message communication between PNs and that between APs are expressed by only MPI. Hence, in this paper the p-mode HDDM is applied to the Earth Simulator but in the future, h-mode by the hybrid parallelization will be applied.

3.3 Balancing Domain Decomposition (BDD)

The ADVENTURE_Solid has employed the BDD method for solving linearized equations. The BDD algorithm is based on the DDM with a preconditioned iterative solver. In the DDM, the analysis domain is decomposed into non-overlapping subdomain, and then a generated artificial boundary among subdomains is called interface. After eliminating interior DOF of local subdomain matrices, the problem to be solved is reduced onto the interface DOF. The reduced matrix is so-called Schur complement. The reduced problem is also called the interface problem, and is to be solved by a preconditioned iterative method. There are two main methods as such preconditioner, i.e. local subdomain correction and coarse grid correction in a coarse space. Main elemental calculations appeared in the BDD algorithm is described below:

a) $\boldsymbol{Ku} = \boldsymbol{f}$

is a linear algebraic equation to be solved, where $\boldsymbol{K}$ is the global stiffness matrix, $\boldsymbol{u}$ is the nodal displacement vector, and $\boldsymbol{f}$ is the external force vector. In the elastic problem, the stiffness matrix $\boldsymbol{K}$ is symmetric positive definite.

b) $\boldsymbol{Su_B} = \boldsymbol{g}, \quad \boldsymbol{S} = \sum_{i=1}^{N} \boldsymbol{R}_i^T \boldsymbol{S}_i \boldsymbol{R}_i$

is the reduced system, where $\boldsymbol{S}$ is the Schur complement for the global stiffness matrix, $\boldsymbol{u_B}$ is the displacement vector on the interface DOF, and $\boldsymbol{g}$ is the given vector. Here, $\boldsymbol{S}$ is symmetric positive definite, and $\boldsymbol{S_i}$ is the local Schur complement of subdomain $\boldsymbol{i} = 1,\ldots,\boldsymbol{N}$, assumed to be positive semi-definite. At first the reduced system on the interface DOF is solved, secondary the local problems on the interior DOF of each subdomain are solved, and then the whole DOF can be known consequently.

c) $\boldsymbol{R}_i, \ \boldsymbol{R}_i \boldsymbol{R}_i^T = \boldsymbol{I}, \ \boldsymbol{i} = 1,\ldots, \boldsymbol{N}$

is the matrix of the global to the local DOF corresponding to interface mapping for subdomain $\boldsymbol{i}$. $\boldsymbol{I}$ is the unit matrix that have the interface DOF.

d) $\boldsymbol{D_i}, \ \sum_{i=1}^{N} \boldsymbol{R}_i^T \boldsymbol{D}_i \boldsymbol{R}_i = \boldsymbol{I}$

is a weighting matrix for subdomain $\boldsymbol{i}$, assumed to form decomposition of unity.

e) $\boldsymbol{Z_i}, \ \boldsymbol{Null\ S_i} \subset \boldsymbol{Range\ Z_i}, \ \boldsymbol{i} = 1,\ldots, \boldsymbol{N}$

is the local coarse space of subdomain $\boldsymbol{i}$, that contains all potential local singularities. Here, $\boldsymbol{Null\ S_i}$ is the null space of the local Schur complement of subdomain $\boldsymbol{i}$.

f) $\boldsymbol{R_W}, \ \boldsymbol{R_W}^T = \left[\boldsymbol{R}_1^T \boldsymbol{D}_1 \boldsymbol{Z}_1, \ldots, \boldsymbol{R}_N^T \boldsymbol{D}_N \boldsymbol{Z}_N \right]$

is the weighted restriction from the global to the coarse DOF.

g) $\boldsymbol{P}, \ \boldsymbol{P} = \boldsymbol{R}_W^T \left(\boldsymbol{R}_W \boldsymbol{S} \boldsymbol{R}_W^T \right)^{-1} \boldsymbol{R}_W \boldsymbol{S}$

is the S-orthogonal projection onto the coarse space. Here, $\boldsymbol{S_W} = \boldsymbol{R_W} \boldsymbol{S} \boldsymbol{R}_W^T$ is a coarse grid operator, assumed to be positive definite.

Various domain decomposition methods contain a process of solving a reduced system using iterative methods such as the preconditioned CG method. At each iterative step, the DDM or the HDDM requires to solve the following auxiliary problem:

$\boldsymbol{Mz} = \boldsymbol{r}$,

where $\boldsymbol{M}$ is a symmetric positive definite matrix called preconditioner and $\boldsymbol{r}$ is a residual vector in each iterative step. The BDD preconditioned operator is described [10, 12] by:

$$\boldsymbol{M}_{BDD}^{-1} \boldsymbol{S} = \boldsymbol{P} + \left(\boldsymbol{I} - \boldsymbol{P} \right) \left(\sum_{i=1}^{N} \boldsymbol{T}_i \right) \left(\boldsymbol{I} - \boldsymbol{P} \right)^T \quad ,$$

where $\boldsymbol{T_i}$ is the local subdomain correction of subdomain $\boldsymbol{i}$, and $\boldsymbol{I}$–$\boldsymbol{P}$ is the coarse grid correction. If $\boldsymbol{P^T r} = 0$, which means a residual vector has no components of the coarse space, the BDD preconditioned operator can be simplified as:

$$\boldsymbol{M}_{BDD}^{-1} \boldsymbol{S} = \left(\boldsymbol{I} - \boldsymbol{P} \right) \left(\sum_{i=1}^{N} \boldsymbol{T}_i \right) \quad ,$$

The original BDD employs the N-N type algorithm as local subdomain correction with a two-level weighted sum of the inverses of $\boldsymbol{S_i}$ matrices [10]. To calculate the inverse of them, the Moore-Penrose pseudo-inverse or some regularization is required since $\boldsymbol{S_i}$ matrices are typically singular. However the Moore-Penrose pseudo-inverse takes high computational cost, while the regularization is less accurate. To overcome those issues of computation cost and accuracy simultaneously, we choose the diagonal scaling preconditioner for $\boldsymbol{S_i}$ as local subdomain correction. As the local subdomain correction based on the diagonal scaling can be applied subdomain-wise, its parallel algorithm is basically compatible to the HDDM.

In the parallelism of the BDD, the importance issue is how to parallelize the operation of the coarse grid correction. The coarse grid correction is applied to solve a linear system equation whose coefficient matrix is a coarse grid operator. Now, a coarse grid operator generally becomes sparse matrix, and the coarse grid correction is implemented in each iteration with its own right hand side vector. To save a computational time, a Cholesky

Table 1 Runtime performances with the original code in the case of 60 subdomains

Function name	MFLOPS	V.OP RATIO	AVER.V.LEN
SKY_AddMultVec	657.0	93.59	26.3
SKY_MkSollution	775.5	93.66	134.0
SKY_Decomposite	457.6	91.45	123.0

Table 2 Runtime performances with the improved code

Subdomains	Function name	MFLOPS	V.OP RATIO	AVER.V.LEN
60	SKY_AddMultVec	1,444.7	96.88	144.6
	SKY_MkSollution	977.3	94.78	139.3
	SKY_Decomposite	1,165.4	97.95	169.8
20	SKY_AddMultVec	2,107.8	97.74	170.8
	SKY_MkSollution	1,459.4	96.71	168.1
	SKY_Decomposite	1,503.6	98.29	185.4
8	SKY_AddMultVec	2,947.3	98.40	196.3
	SKY_MkSollution	2,090.8	97.86	194.0
	SKY_Decomposite	1,961.4	98.62	200.8
3	SKY_AddMultVec	3,189.2	98.55	202.9
	SKY_MkSollution	2,247.3	98.07	199.2
	SKY_Decomposite	2,041.2	98.68	203.2

```
  a[1] = 1.0/a[1];
  for (iDof = 2; iDof <= nDofs; iDof++) {
    int ifst=minIndexes[iDof];
    int ii=diagIndexes[iDof]-iDof;
#pragma cdir nodep
    for (jDof = ifst; jDof < iDof; jDof++) {
      if (ifst > minIndexes[jDof])
        WORKDATA[jDof] = ifst;
      else
        WORKDATA[jDof] = minIndexes[jDof];
    }
    for (jDof = ifst; jDof < iDof; jDof++) {
      int jj=diagIndexes[jDof]-jDof;
#pragma cdir nodep
      for (kDof = WORKDATA[jDof]; kDof < jDof; kDof++)
        a[ii+jDof]-=a[jj+kDof]*a[ii+kDof];
    }
#pragma cdir nodep
    s = 0.0;
    for (jDof = ifst; jDof < iDof; jDof++) {
      s += a[ii+jDof]*a[ii+jDof]*a[diagIndexes[jDof]];
      a[ii+jDof] *= a[diagIndexes[jDof]];
    }
    a[diagIndexes[iDof]]=1.0/(a[diagIndexes[iDof]]-s);
  }
```

Fig. 7 Original code for LDL decomposition in the ADVENTURE_Solid

factorized coarse grid operator in the first iteration step is kept, and then the forward substitution and the backward substitution of a coarse space problem are only implemented after the second iteration step. To realize this technique, the ADVENTURE_Solid employs a parallel direct sparse solver.

4. Tuning for the Earth Simulator

The ADVENTURE_Solid has shown excellent performances not only on PC clusters consisting of scalar processors, but also on MPPs such as Hitachi SR8000/MPP consisting of 1,024 pseudo-vector processors without changing its code. This section describes the implementation of the ADVENTURE_Solid onto the Earth Simulator with improvement of some functions for vector-type processors.

The principal indices of performance on MPP are FLOPS and parallel efficiency. To attain a high performance of FLOPS, vector operation ratio (V.OP RATIO) and average of vector length (AVER.V.LEN) are key issues. To attain a high parallel efficiency, parallel architecture is a key issue. In the present study, we tune the

```
#pragma cdir nodep
  for (iDof = 0; iDof < nDofs; iDof++)
    aDiag[iDof] = a[Index_ij[iDof]+iDof];
  for (iDof = 0; iDof < nDofs; iDof+=3) {
    int ii0=Index_ij[iDof];
    int ii1=Index_ij[iDof+1];
    int ii2=Index_ij[iDof+2];
    double r0_0=0.0;
    double r1_0=0.0;
    double r2_0=0.0;
    double r1_1=0.0;
    double r2_1=0.0;
    double r2_2=0.0;
#pragma cdir nodep
    for (jDof = Index_ii[iDof]; jDof < iDof; jDof+=3) {
      if (Index_ii[iDof] > Index_ii[jDof])
        WORK[jDof] = Index_ii[iDof];
      else
        WORK[jDof] = Index_ii[jDof];
    }
    for (jDof = Index_ii[iDof]; jDof < iDof; jDof+=3) {
      double s0_0=0.0;
      double s1_0=0.0;
      double s2_0=0.0;
      double s0_1=0.0;
      double s1_1=0.0;
      double s2_1=0.0;
      double s0_2=0.0;
      double s1_2=0.0;
      double s2_2=0.0;
      int jj0=Index_ij[jDof];
      int jj1=Index_ij[jDof+1];
      int jj2=Index_ij[jDof+2];
#pragma cdir nodep
      for (kDof = WORK[jDof]; kDof < jDof; kDof++) {
        s0_0 += a[jj0+kDof]*a[ii0+kDof];
        s0_1 += a[jj0+kDof]*a[ii1+kDof];
        s0_2 += a[jj0+kDof]*a[ii2+kDof];
        s1_0 += a[jj1+kDof]*a[ii0+kDof];
        s1_1 += a[jj1+kDof]*a[ii1+kDof];
        s1_2 += a[jj1+kDof]*a[ii2+kDof];
        s2_0 += a[jj2+kDof]*a[ii0+kDof];
        s2_1 += a[jj2+kDof]*a[ii1+kDof];
        s2_2 += a[jj2+kDof]*a[ii2+kDof];
      }
      a[ii0+jDof] -= s0_0;
      a[ii1+jDof] -= s0_1;
      a[ii2+jDof] -= s0_2;
      s1_0 += a[jj1+jDof]*a[ii0+jDof];
      s1_1 += a[jj1+jDof]*a[ii1+jDof];
      s1_2 += a[jj1+jDof]*a[ii2+jDof];
      a[ii0+jDof+1] -= s1_0;
      a[ii1+jDof+1] -= s1_1;
      a[ii2+jDof+1] -= s1_2;
      s2_0 += a[jj2+jDof]*a[ii0+jDof]+a[jj2+jDof+1]*a[ii0+jDof+1];
      s2_1 += a[jj2+jDof]*a[ii1+jDof]+a[jj2+jDof+1]*a[ii1+jDof+1];
      s2_2 += a[jj2+jDof]*a[ii2+jDof]+a[jj2+jDof+1]*a[ii2+jDof+1];
      a[ii0+jDof+2] -= s2_0;
      a[ii1+jDof+2] -= s2_1;
      a[ii2+jDof+2] -= s2_2;
    }
#pragma cdir nodep
    for (kDof = Index_ii[iDof]; kDof < iDof; kDof++) {
      kij0 = a[ii0+kDof]*aDiag[kDof];
      kij1 = a[ii1+kDof]*aDiag[kDof];
      kij2 = a[ii2+kDof]*aDiag[kDof];
      r0_0 += a[ii0+kDof]*kij0;
      r1_0 += a[ii1+kDof]*kij0;
      r1_1 += a[ii1+kDof]*kij1;
      r2_0 += a[ii2+kDof]*kij0;
      r2_1 += a[ii2+kDof]*kij1;
      r2_2 += a[ii2+kDof]*kij2;
      a[ii0+kDof] = kij0;
      a[ii1+kDof] = kij1;
      a[ii2+kDof] = kij2;
    }
    a[ii0+iDof]=1.0/(a[ii0+iDof]-r0_0);
    aDiag[iDof]=a[ii0+iDof];
    a[ii1+iDof] -= r1_0;
    a[ii2+iDof] -= r2_0;
    kij0 = a[ii1+iDof]*aDiag[iDof];
    r1_1 += a[ii1+iDof]*kij0;
    a[ii1+iDof] = kij0;
    a[ii1+iDof+1]=1.0/(a[ii1+iDof+1]-r1_1);
    aDiag[iDof+1]=a[ii1+iDof+1];
    a[ii2+iDof+1] -= r2_1 + a[ii1+iDof]*a[ii2+iDof];
    kij0 = a[ii2+iDof]*aDiag[iDof];
    kij1 = a[ii2+iDof+1]*aDiag[iDof+1];
    r2_2 += a[ii2+iDof]*kij0 + a[ii2+iDof+1]*kij1;
    a[ii2+iDof] = kij0;
    a[ii2+iDof+1] = kij1;
    a[ii2+iDof+2]=1.0/(a[ii2+iDof+2]-r2_2);
    aDiag[iDof+2]=a[ii2+iDof+2];
  }
```

Fig. 8 Improved code for LDL decomposition in the ADVENTURE_Solid

code focusing on operations of local subdomain analysis, which are most influential to FLOPS performance.

The HDDM involves the following three main calculations: (1) LDL factorization of subdomain's coefficient matrices (denoted as SKY_Decomposite), (2) forward elimination and backward substitution of linear equations (denoted as SKY_MkSolution), and (3) matrix multiplication by a vector (denoted as SKY_AddMultVec). These functions are vectorized, respectively. Table 1 shows runtime performance, FLOPS, V.OP RATIO and AVER.V.LEN with the original code when applied to a test model, which has about 23,000 DOF and decomposed into 60 subdomains. As the original code is programmed for scalar processors, V.OP RATIO stays in a lower level and AVER.V.LEN is short. It should be also noted here that the number of subdomains is selected so as to be suitable to scalar processors.

In the present implementation, we improve the code for the Earth Simulator based on the following policy: (1) simple coding for easy compiling, (2) changing the innermost loop to continuous array access, and (3) unrolling of the outer loop by 3 times because each node of a solid element has 3 DOF in three-dimensional solid problems. As for AVER.V.LEN, we can get longer loop length by changing the number of subdomains. Table 2 shows runtime performance with the improved code measured for a test model, which is decomposed into several numbers of subdomains. In the table, the runtime performances are improved well. Furthermore, with the decreasing number of subdomains, the runtime performances are more improved because of increasing the vector length. Especially, in the case of 3 subdomains (7,667 DOF/subdomain), it shows good performance of over 25% to peak FLOPS of a single AP, i.e. 8 GFLOPS and over 98% performance for vectorization. Fig. 7 and Fig. 8 show examples of original code and improved one, respectively.

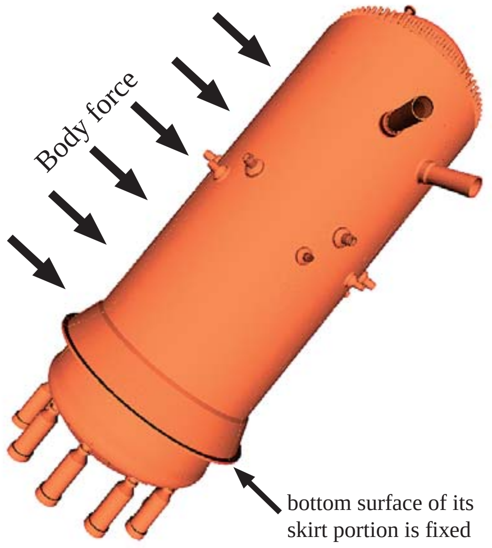

Fig. 9 The analysis conditions for elastostatic analysis of the ABWR model

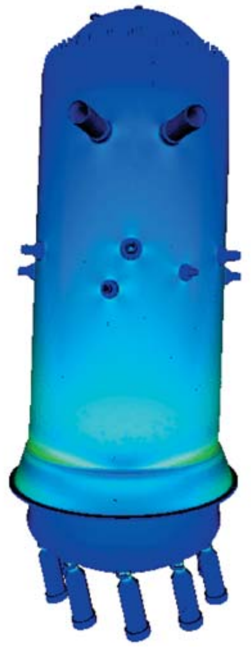

Fig. 10 Stress distribution and deformed shape of the ABWR model with 35 million DOF

5. Numerical Experiments

5.1 Elastostatic Analysis of ABWR Vessel Model with 35 Million DOF

This section describes an elastostatic stress analysis for a precise model of an advanced boiling water reactor (ABWR) vessel with a 35 million DOF unstructured mesh as shown in Fig. 2. As boundary conditions, the bottom surface of its skirt portion is fixed, and a static gravitational force is applied to the vessel in the horizontal direction, imitating a seismic loading condition. These analysis conditions are illustrated in Fig. 9. Its mesh size and total DOF of analysis model are listed in Table 3. Such a complex shaped and large-scale thin structure with less constraint often results in an ill-conditioned system matrix. Most iterative solution methods suffer from poor convergence when solving such ill-conditioned problems. However the ADVENTURE_Solid overcomes this problem owing to the employment of the BDD method. Fig. 10 shows the calculated stress distribution and deformed shape.

Fig. 11 shows convergence histories of force imbalance measured at the interface of subdomains, i.e. residual norm, plotted against the number of iterations. The stopping criterion is that the norm of the relative residual is reduced to 1.0e-6, and then all of the following problems employ the same stopping criterion. The calcula-

Table 3 Mesh size of the ABWR model with 35 million DOF

Number of elements	Number of nodes	Total degrees of freedom
7,486,792	11,794,506	35,368,551

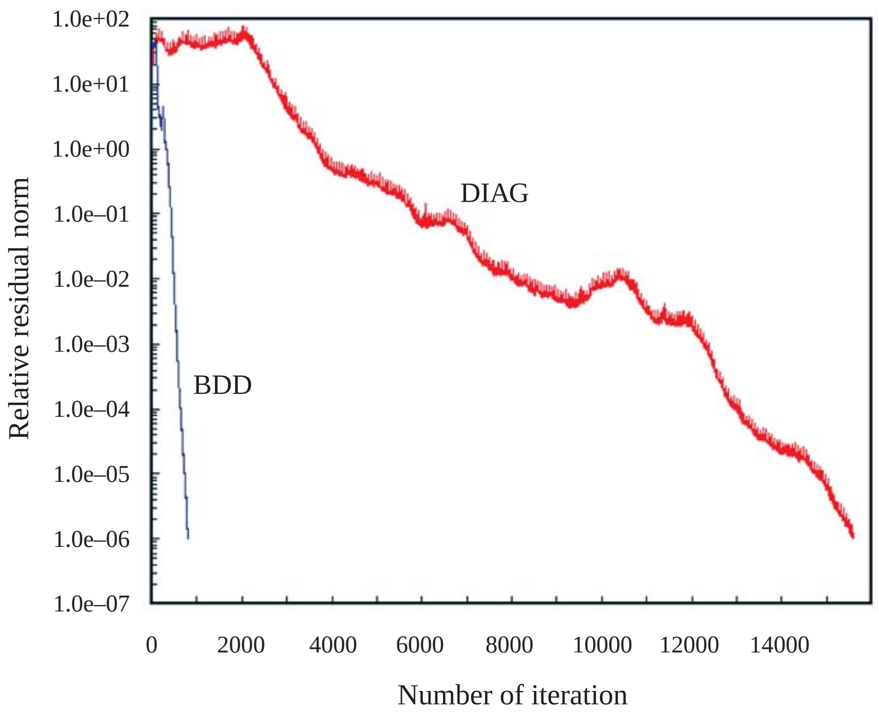

Fig. 11 Comparison with DIAG and BDD of the history of relative residual norm of elastostatic analysis of the ABWR model

tions are performed on the Earth Simulator consisting of 128 nodes (1,024 PEs) with 2TB of main memory, whose theoretical peak performance is 8 TFLOPS. There are two lines in the figure. DIAG denotes the result obtained using the HDDM with a diagonal-scaling preconditioner, while BDD does that of the HDDM with the BDD preconditioner. In the DIAG case, the number of iteration until the convergence is 17,564 in 46.8 minutes. On the other hand, in the BDD case, that is 852 in 8.1 minutes. As the results, the BDD successfully reduces the number of iterations to less than 5% and the computational time to less than 20% compared with the DIAG. With consideration to unsteady and nonlinear analysis for a real world problem, the speed-up of linear solver is the great importance matter.

We evaluated the practicality of parallel computing, varying the number of processors employed. For the purpose of comparison, such evaluation was performed using both the Hitachi SR8000/MPP at the Supercomputer Center of the University of Tokyo and the Earth Simulator. The results are shown in Figs. 12–15. As shown in Fig. 12 from the results obtained using the SR8000/MPP, parallel efficiency over the whole calculation using 1,024 PEs was estimated as about 20%, referring the value for using 128 PEs. This is attributed due to the fact that convergence feature of the BDD method of the ADVENTURE_Solid depends on the number of processors employed. For instance, the 128 PEs case and 1,024 PEs case require 476 and 1,116 iteration counts, respectively. On the other hand, parallel performance for one iterative calculation was estimated as about 60% from Fig. 13. It is judged from such high parallel efficiency that the present system is sufficiently parallelized. More speed-up is expected by improving the convergence performance in the case of the larger number of processors. As shown in Fig. 14 and Fig. 15, the similar tendency is observed for the case using the Earth Simulator. However, the parallel efficiency over the whole calculation was about 53% from Fig. 14. Such better performance on the Earth Simulator is attributed due to much better computing performance and wider band width of communication of the Earth Simulator.

5.2 Seismic Response Analysis of ABWR Vessel Model with 35 Million DOF

The second problem is a seismic response analysis of

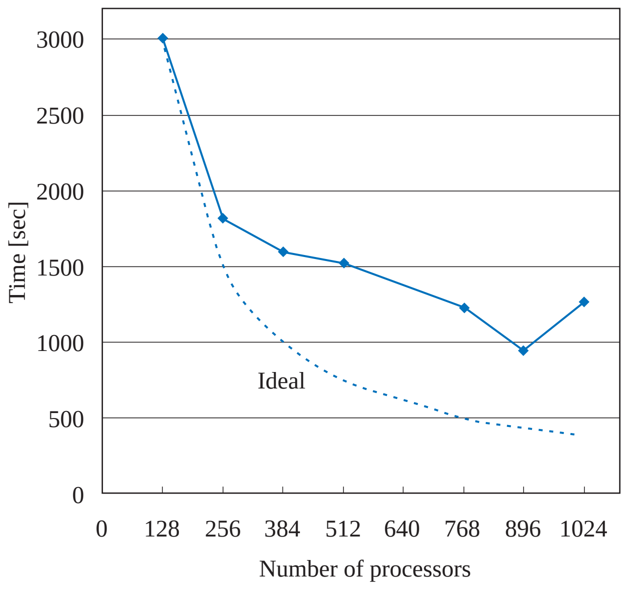

Fig. 12 Scalability in total computation time of elastostatic analysis of the ABWR model on the Hitachi SR8000/MPP

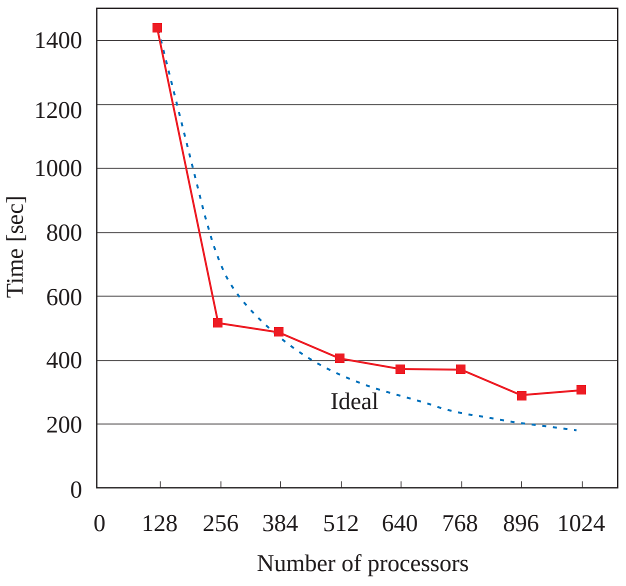

Fig. 14 Scalability in total computation time of elastostatic analysis of the ABWR model on the Earth Simulator

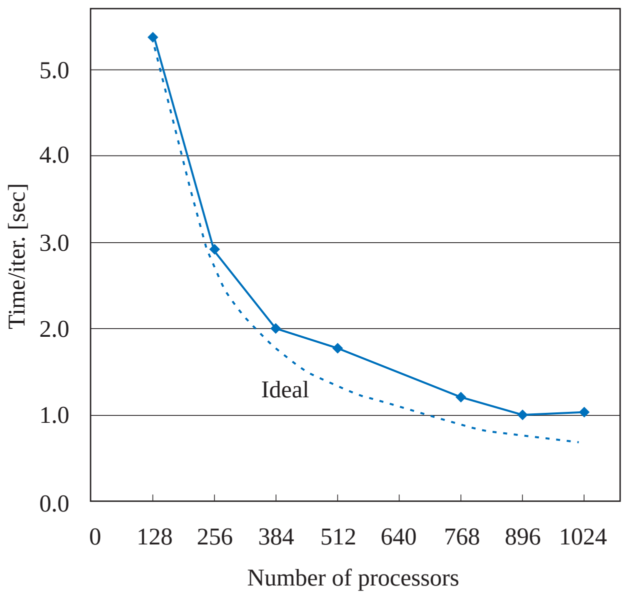

Fig. 13 Scalability in total computation time per iteration of elastostatic analysis of the ABWR model on the Hitachi SR8000/MPP

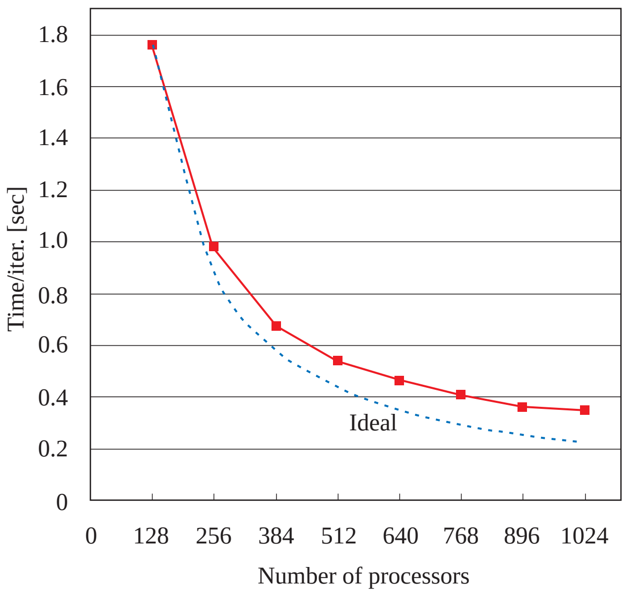

Fig. 15 Scalability in total computation time per iteration of elastostatic analysis of the ABWR model on the Earth Simulator

the ABWR model using the Earth Simulator consisting of 128 nodes (1,024 PEs). As boundary conditions, a bottom plane of its skirt portion is fixed same as the previous section. As a seismic load, the acceleration history of 1,940 Elcentro earthquake ground motion is taken, whose data is provided by the Building Center of Japan [29]. Fig. 16 plots the time history of the acceleration, whose time delta is 0.02 seconds. Fig. 17 and Fig. 18 show the stress distribution and the deformed configuration in 40 and 80 time steps, respectively.

Fig. 19 shows the histories of residual norm, solved with the DIAG solver, and with the BDD solver. The BDD succeeded in calculation time about 6 times speed-up compared with DDM. Table 4 shows the comparison with DIAG and BDD solver of the calculation performances in the first three steps. In the seismic response analysis, that is elastic dynamic analysis, the BDD also successfully reduces the number of iterations to less than 1% and the computational time to less than 7% compared with the DIAG. Table 5 shows the calculation perform-

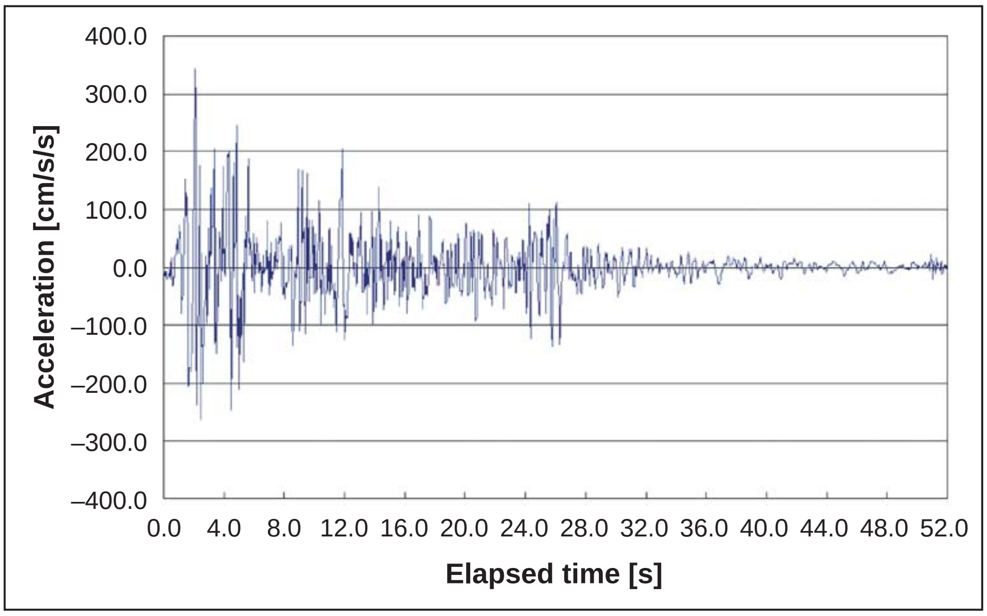

Fig. 16 ElCentro earthquake ground motion 1940, NS direction, provided by the Building Center of Japan

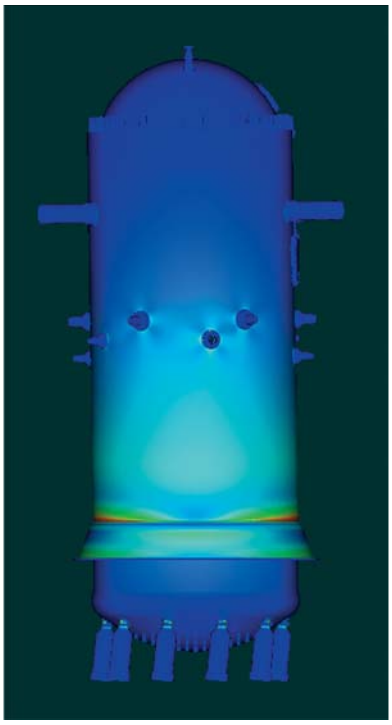

Fig. 17 Stress distribution and deformed configuration of the ABWR model, after elapsed 0.8 seconds, 40 time steps

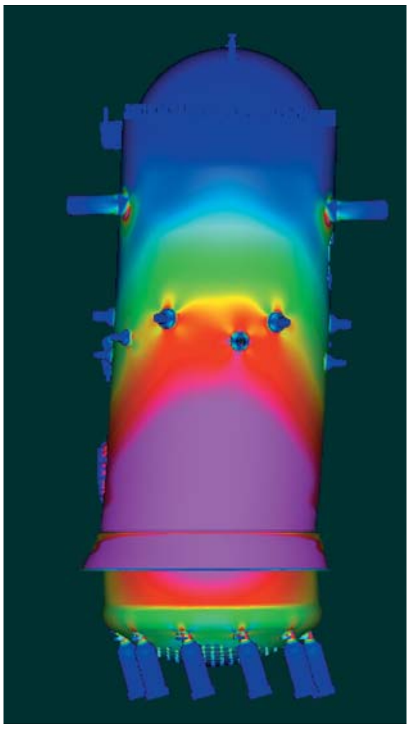

Fig. 18 Stress distribution and deformed configuration of the ABWR model, after elapsed 1.6 seconds, 80 time steps

ances of 300 steps by BDD. As shown in the table, the present system is successfully to analyze a seismic response analysis of a precise and whole structure model of 300 time steps in 4.3 hours.

5.3 Elastostatic Analysis of Pressure Vessel Model with 100 Million DOF

The present system is applied to an elastostatic stress analysis of a pressure vessel model with 100 million DOF mesh. Its mesh size is listed in Table 6. As boundary conditions, the bottom surface of the vessel is fixed, and a static gravitational force is applied to the vessel in the

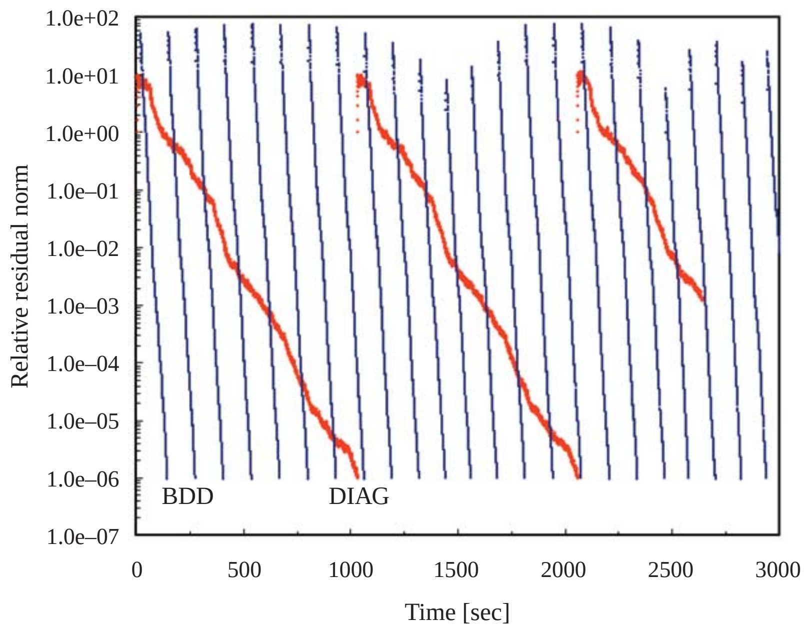

Fig. 19 Comparison with DIAG and BDD of the time vs. relative residual norm of seismic response analysis of the ABWR model

Table 4 Comparison with DIAG and BDD of the calculation performance of seismic response analysis of the ABWR model

Solver type	Time step	Number of iterations	Computational time (sec)
DIAG	1st	12,470	1,031.4
	2nd	12,478	1,027.2
	3rd	12,559	1,033.3
BDD	1st	143	69.1
	2nd	143	50.2
	3rd	143	50.2

Table 5 Calculation performances of seismic response analysis of the ABWR model by BDD solver

Time step	Computational time (hr) (total)	GFLOPS	Memory (GB)
1–100	1.43 (1.43)	960	647
101–200	1.43 (2.86)	955	647
201–300	1.44 (4.30)	957	647

Table 6 Mesh size of a pressure vessel model with 100 million DOF

Number of elements	Number of nodes	Total degrees of freedom
25,084,456	34,772,634	104,195,500

horizontal direction. These analysis conditions are illustrated in Fig. 20. Fig. 21 shows the calculated stress distribution and deformed shape.

The present system with the BDD is successfully to analyze with good performances in FLOPS and vectorization as shown in Table 7. In the DDM algorithm, the number of subdomains affects performance of the calculation. Even in the same model, the optimized number of subdomains depends on the computer environments, i.e. number of processors or amount of memory. In this case, the optimized number of subdomains is about 6,000 on the 2,048 PEs. With such optimized number of subdo-

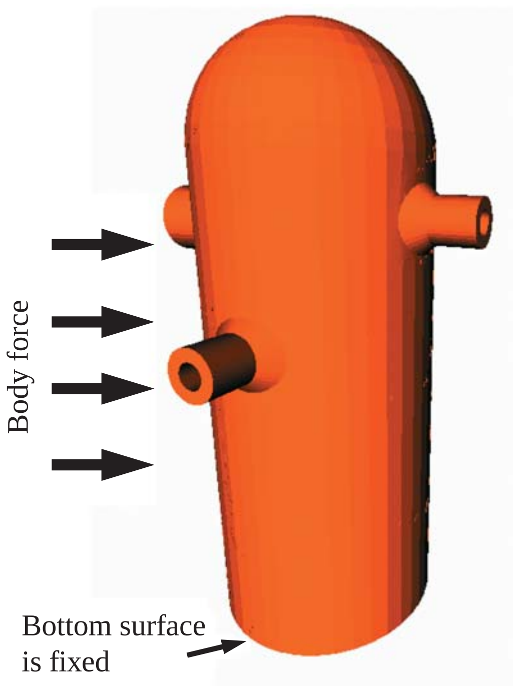

Fig.20 The analysis conditions for elastostatic analysis of the pressure vessel model with 100 million DOF

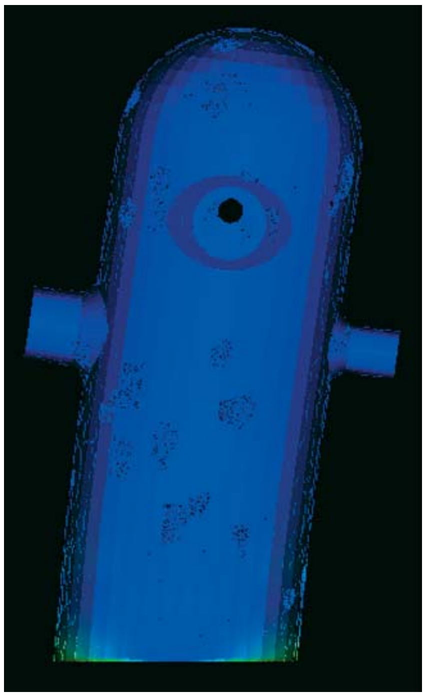

Fig. 21 Stress distribution and deformed shape of the pressure vessel model with 100 million DOF

Table 7 Calculation performances of 100 million DOF problem by BDD solver

# PEs	Subdomains	Time (sec)	Iterations	TFLOPS	(to peak %)	V.OP RATIO (%)
512	34,816	1743	1,074	1.12	(28.0)	98.1
1,024	34,816	960	1,038	1.97	(24.6)	98.0
2,048	34,816	835	1,330	2.88	(18.0)	97.9
2,048	6,144	514	496	5.08	(31.75)	98.8

mains, the present system performed better vectorization, and then successfully archived 5.08 TFLOPS, which is 31.75% of peak performance. As the result, the elastostatic analysis of pressure vessel model with 100 million DOF is successfully performed in 8.6 minutes.

6. Conclusions

We have been developing an advanced general-purpose finite element analysis system, named ADVENTURE, which is designed to be able to analyze a model of arbitrary shape over 100 million DOF mesh. The ADVENTURE_Solid has been successfully implemented on a single PC, PC clusters and massively parallel processors. In this paper, this solid analysis solver is implemented on the Earth Simulator consisting of 256 nodes (2,048 PEs) with theoretical peak performance of 16 TFLOPS, and succeeds in solving an elastostatic problem of a nuclear pressure vessel model of 100 million DOF with 5.08 TFLOPS, which is 31.75% of the peak performance. The ADVENTURE system on the Earth Simulator is to be used for virtual mockup tests of large-scale and complex artifacts such as nuclear pressure vessels subjected to seismic loading.

Acknowledgments

The authors would like to thank the Earth Simulation Center for giving us the opportunity to use the Earth simulator. The authors also would like to thank the Supercomputer Center of the University of Tokyo for giving us the opportunity to use the Hitachi SR8000/MPP. This work is performed as a part of the ADVENTURE project and the authors also would like to thank all the members of the ADVENTURE project.

(This article is reviewed by Dr. Tetsuya Sato.)

References

[1] http://adventure.q.t.u-tokyo.ac.jp
[2] S. Yoshimura, R. Shioya, H. Noguchi and T. Miyamura, Advanced general-purpose computational mechanics sys-

tem for large-scale analysis and design, *Journal of Computational and Applied Mathematics*, vol.**49**, pp.279–296, 2000.

[3] Report on Computational Science and Engineering, JSPS-RFTF Program 2001-2002, 2002.

[4] http://proton.is.s.u-tokyo.ac.jp/cse/index-e.html/

[5] M. Suzuki, T. Ohyama, H. Akiba, H. Noguchi and S. Yoshimura, Development of fast and robust parallel Coarse-grid based CG solver for large scale finite element analyses, *Transactions of Japan Society of Mechanical Engineers*, vol.**68A–671**, pp.1010–1017, 2002.

[6] G. Yagawa and R. Shioya, Parallel finite elements on a massively parallel computer with domain decomposition, *Computing Systems in Engineering*, vol.**4**, pp.495–503, 1994.

[7] R. Shioya and G. Yagawa, Parallel finite element analysis of 100 million DOF based on the hierarchical domain decomposition method, *Transactions of Japan Society of Computational Engineering and Science*, vol.**3**, pp.201–206, 2001 (in Japanese).

[8] T. Miyamura, H. Noguchi, R. Shioya, S. Yoshimura and G. Yagawa, Elastic-plastic analysis of nuclear structures with millions of DOF using the hierarchical domain decomposition method, *Nuclear Engineering & Design*, vol.**212**, pp.335–355, 2002.

[9] Y. H. De Roeck and P. LeTallec, Analysis and test of a local domain decomposition preconditioner, *Fourth International Symposium on Domain Decomposition Methods*, pp.112–128, 1991.

[10] J. Mandel, Balancing domain decomposition, *Communications on Numerical Methods in Engineering*, vol.**9**, pp.233–241, 1993.

[11] P. LeTallec and M. Vidrascu, Generalized neumann-neumann preconditioners for iterative substructuring, *Ninth International Symposium on Domain Decomposition Methods*, pp.413–425, 1996.

[12] P. LeTallec, J. Mandel and M. Vidrascu, A neumann-neumann domain decomposition algorithm for solving plate and shell problems, *SIAM J. Number. Math.*, vol.**35**, pp.836–867, 1997.

[13] R. Shioya, M. Ogino, H. Kanayama and D. Tagami, Large scale finite element analysis with a balancing domain decomposition method, *Key Engineering Materials*, vol.**243–244**, pp.21–26, 2003.

[14] R. Shioya, H. Kanayama, A.M.M. Mukaddes and M. Ogino, Heat conductivity analyses with balancing domain decomposition, *Theoretical and Applied Mechanics Japan*, vol.**52**, pp.43-53, 2003.

[15] T. Miyamura and S. Yoshimura, Parallel stress analyses of ancient architecture Pantheon on PC cluster, *Transactions of Architectural Institute of Japan*, vol.**55**, pp. 95–102, 2001 (in Japanese).

[16] S. Yoshimura, H. Nitta, G. Yagawa and H. Akiba, Parallel automatic mesh generation of nuclear structures with ten-million nodes, *Transactions of 15-th International Conference on Structural Mechanics in Reactor Technology, Seoul*, vol.**II**, pp.21–28, 1999.

[17] G. Karypis and V. Kumar, Multilevel k-way partitioning scheme for irregular graphs, *Technical Report TR 95-064, Department of Computer Science, University of Minnesota*, 1995.

[18] G. Karypis and V. Kumar, Parallel multilevel k-way partitioning scheme for irregular graphs, *Technical Report TR 96-036, Department of Computer Science, University of Minnesota*, 1996.

[19] H. Kanayama, R. Shioya, D. Tagami and H. Zheng, A numerical procedure for 3-D nonlinear magetostatic problems using the magnetic vector potential, Theoretical and Applied Mechanics, Vol.**50**, pp.411–418, 2001.

[20] S. Shoui, S. Yoshimura, H. Akiba, T. Ohyama and G. Yagawa, Parallel visualization of finite element solutions with ten million DOF using PC cluster, *Proceedings of European Congress on Computational Methods in Science and Engineering (ECCOMAS2000), Balcelona*, CD-ROM, 2000.

[21] M. Vidrascu, Remarks on the implementation of the generalized neumann-neumann algorithm, *11th International Conference on Domain Decomposition Methods*, pp.485–493, 1998.

[22] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and P. Plechac, PARASOL. An integrated programming environment for parallel sparse matrix solvers, *High-Performance Computing*, pp.79–90, 1999.

[23] P. Goldfeld, Balancing neumann-neumann for (in)compressible linear elasticity and (generalized) stokes - parallel implementation, *14th International Conference on Domain Decomposition Methods*, 2002.

[24] F.-X. Roux and C. Farhat, Parallel implementation of the two-level feti method, *9th International Conference on Domain Decomposition Methods*, pp.480–487, 1997.

[25] J. Mandel, R. Tezaur and C. Farhat, A scalable substructuring method by lagrange multipliers for plate bending problems, *SIAM Journal of Numerical Analysis*, vol.**36**, pp.1370–1391, 1999.

[26] M. Lesoinne and K. Pierson, FETI-DP: An efficient, scalable and unified dual-primal feti method, *12th International Conference on Domain Decomposition Methods*, pp.421–428, 1999.

[27] http://endo.sandia.gov/9234/salinas

[28] M. Bhardwaj, K. Pierson, G. Reese, T. Walsh, D. Day, K. Alvin, J. Peery, C. Farhat, and M. Lesoinne, Sanlias: A scalable software for high-performance structural and solid mechanics simulations, *Technical Papers of SC2002*, 2002.

[29] http://www.bcj.or.jp/