WCCM V
Fifth World Congress on
Computational Mechanics
July 7-12, 2002, Vienna, Austria
Eds.: H.A. Mang, F.G. Rammerstorfer,
J. Eberhardsteiner

Generalized Approach to Parallel Shape Optimization with Millions DOF Finite Element Model

Shinobu Yoshimura*, Brian Dennis, and Hiroshi Kawai

Institute of Environmental Studies, The University of Tokyo
7-3-1 Hongo, Bunkyo, Tokyo 113-8656, Japan
e-mail: yoshi@k.u-tokyo.ac.jp

Tomonari Furukawa

School of Aerospace, Mechanical and Mechatronic Engineering
University of Sydney, NSW 2006, Australia
e-mail: tomo@acfr.usyd.edu.au

Key words: shape optimization, parallel processing, hierarchical domain decomposition, module-based architecture, I/O standardization

Abstract
We have been developing an advanced general-purpose computational mechanics system, named ADVENTURE, which is designed to be able to analyze a model of arbitrary shape with 10 million to 100 million degrees of freedom (DOFs) mesh, and additionally to enable optimization. Domain-decomposition based parallel algorithms are implemented in pre-processes (domain decomposition), main processes (system matrix assembling and solutions) and post-process (visualization), respectively. The hierarchical domain decomposition method with a preconditioned iterative solver (HDDM) is adopted in the main processes as one of major solution techniques. Module-based architecture of system with standardized I/O format and libraries are also developed and employed to attain flexibility, portability, extensibility and maintainability of the whole system. This paper first describes a whole view of the ADVENTURE system, and then explain some optimization algorithms and their implementation into the system. For non-parametric shape optimization, we employ the Traction method. Gradient-based algorithms, genetic and evolutionary algorithms are employed for parametric optimization. Finally we demonstrate how one can solve optimization problems of large scale models in an efficient manner using the generalized approach which simply combines the fully automated parallel analysis capabilities of large scale model and the optimization algorithms.

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, can not be used with massively parallel processors (MPPs) with the order of 100-10,000 processing elements (PEs), which are about to dominate 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 (DOFs) at most.
The ADVENTURE project [1-3] 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) [2, 4]. In the project we have been developing an advanced general-purpose computational mechanics system named ADVENTURE since Aug. 1997. The system is designed to be able to analyze a three-dimensional (3D) finite element model of arbitrary shape with 10 million to 100 million DOF mesh, and additionally to enable parametric and non-parametric shape optimization.
In this paper, we first describe a whole view of the ADVENTURE system, and then explain some optimization algorithms and their implementation into the system. For non-parametric shape optimization, we employ the Traction method. Gradient-based algorithms, genetic and evolutionary algorithms are employed for parametric optimization. Finally we demonstrate how one can solve optimization problems of large scale models in an efficient manner using the generalized approach which simply combines the fully automated parallel analysis capabilities of large scale model and the optimization algorithms.

2 System Concept

2.1 Requirements for Next-generation General-purpose Computational Mechanics Systems

When we initiated the ADVENTURE project, we first summarized issues to be solved in next-generation general-purpose computational mechanics systems as follows :

(1) Capability to perform a precise analysis of a whole structure using 10^7-10^8 DOF mesh,
(2) High parallel performance in massively parallel processors (MPPs) with 10^2- 10^4 processing elements (PEs),
(3) Portability among various kinds of parallel and distributed environments ranging from PC cluster to MPPs,
(4) Single and multidisciplinary analyses and design analyses in a parallel and distributed environment in an efficient and unified manner, and
(5) Extensibility and maintainability.

To resolve those issues simultaneously, we decided to develop in the project an advanced general-purpose computational mechanics system for large scale analysis and design, named ADVENTURE.

2.2 Design and Development Strategies

As will be described in Chapter 3, the ADVENTURE system has employed a hierarchical domain-decomposition based massively parallel algorithm as one of major solution algorithms in order to efficiently handle a huge scale finite element model with 10 million to 100 million DOFs. In addition, the following strategies for design and development of the system have been employed :

(1) Module-based architecture,
(2) Standardized I/O format and libraries to collaborate modules in an efficient and unified manner,
(3) Object-oriented design and programming for each module, and
(4) Commodity technologies in hardware as well as software.

Conventional computational mechanics systems have increased their program size to the order of million lines, and become very complicated software through long-histories of program development over 20 years. As the result, it has been a very hard task to even slightly modify their source programs in order to adjust them for version-up of operating system (OS) or to add a little amount of lines for new function. A tremendous amount of efforts have been put to pursue such tasks in daily development. This is one of the main reasons why it is very difficult to develop efficient parallel versions of conventional computational mechanics systems [5].
Even in the case of the ADVENTURE system, it is easily imagined that its total system size would reach to the order of sub-million lines, considering a variety of functions to be implemented in it. Furthermore, yearly version-up of the system would be indispensable, according to yearly version-up of parallel computing environments. Thus, keeping extensibility and maintainability of the system is one of critical issues.
Considering the above aspects, we have decided to employ module-based architecture of the system together with standardized I/O format and libraries [6, 7]. Each portion of the system requiring different background knowledge, theory, algorithm and programming style is packed into an independent module. Figure 1 illustrates the module structure of the ADVENTURE system. The pre-process modules include surface patch generators which convert geometry model data into a collection of triangular surface patch data, a tetrahedral mesh generator [8, 9], an attachment tool of boundary conditions and material properties onto the mesh, and a domain decomposer of a finite element model. The main process modules include an implicit elastic-plastic analysis module [10, 11], a thermal-fluid analysis module [12], a magnetic analysis module [13], an explicit impact analysis module [14, 15], and a rigid plastic analysis module. The post process module is for visualization of analysis results

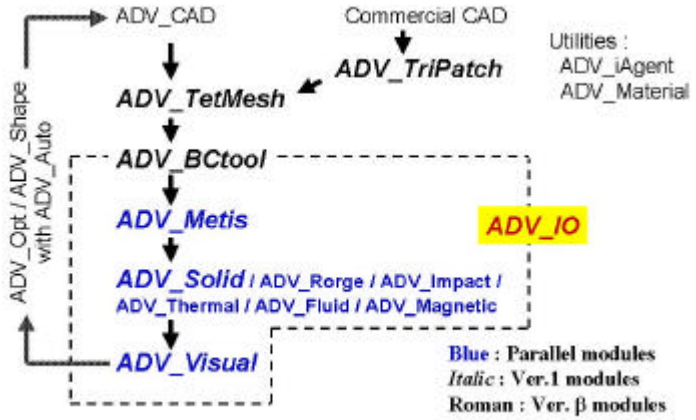

Figure 1 : Module-based architecture of ADVENTURE system.

[16]. Each module is an independent software and its program size is limited to approximately thousands to 30,000 lines, so that each module could be kept manageable by one expert engineer.
There are two main reasons why we newly develop standardized I/O format and libraries. First, different main modules deal with different mechanics variables, while their relevant finite element models must be decomposed into a number of subdomains using a unique domain decomposer module. To perform this process efficiently irrespective of mechanics, it is necessary to standardize I/O format and libraries. Second, the ADVENTURE system is basically required to deal with a large scale of finite element model with 10 million to 100 million DOF mesh in various parallel and distributed computing environments. For this purpose, it is also indispensable to achieve efficient data I/O and data transfer. The developed I/O system is named ADVENTURE_IO, which will be described in detail in the next chapter. Employing the ADVENTURE_IO also provides great benefit to coupled analyses [6, 17] and design optimization calculations, in both of which efficient connection of several different modules is required. This feature will be also explained later.
Other design strategies of the project are to employ only commodity technologies in hardware as well as software, and not to employ any special tools or options. To make R&D processes in software projects efficient, it is usual to employ certain commercial tools, middleware and/or compiling options. However, considering the fact that application programs like CAE systems could have a longer life than those tools and environments, such strategy of employing various tools and special environments might bring a kind of risk into the project. Sometimes license of those tools limits the distribution of developed programs. To avoid such risk as much as possible, we decided to employ only commodity technology in hardware as well as software, and not to employ any special tools or options. In the project, C and C++ are fundamental programming languages to build pre-, main and post-processing modules. MPI is utilized for communication among processors. Operating systems (OS) employed are Unix / Linux. OpenGL and Motif are adopted to deal with graphical data. In the project, we develop a minimum set of GUI and put most efforts to build kernel parts of the system. We expect that some private companies will build high-quality GUI for the ADVENTURE system, considering specific needs for potential applications and users. As described later, it is also essential that if required, all analyses can be performed fully automatically without any intervention by users. To do so, each module is designed to run by command-based operation as well. Although most modules have been developed by us from the start, only one exception is that graph partition tools named METIS and PartMETIS [18, 19], which were developed in University of Minnesota, have been employed as a kernel of the hierarchical domain decomposer of mesh. This is because these are efficient and high-quality software, and furthermore freely used.

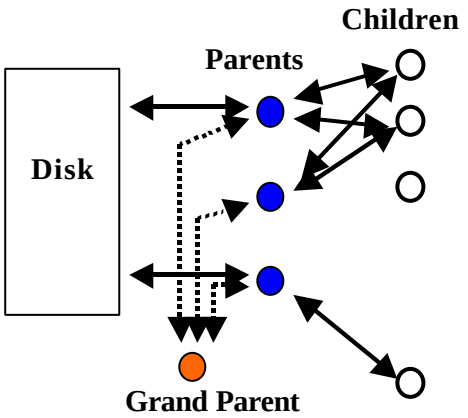

Figure 2 : Data flow among PEs in HDDM.

3 Key Technologies

3.1 Parallel Solver

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. Yagawa and Shioya proposed a hierarchical technique to implement the DDM on MPPs [20]. This technique was named the hierarchical domain decomposition method (HDDM). In the method, a group of processing elements (PEs) are subdivided into three sub-groups : one Grand Parent PE (Grand Parent), several Parent PEs (Parent/Parents), and many Child PEs (Child/Children). For implicitly solving the linear algebraic equations derived from the FEM, a simple substructure-based conjugate gradient (CG) method (the primal substructuring method) [21] is adopted. It was reported in Ref. [21] that convergence property of this method is not so good compared with the dual substructuring methods such as the finite element tearing interconnection (FETI), especially when solving ill-conditioned problems such as large scale thin structures with less constraints. Therefore we have also implemented a preconditioner named Balancing domain decomposition (BDD [22]) into the HDDM [11].
Figure 2 shows the schematic data flow among the three groups of PEs, i.e. Grand Parent, Parents and Children. An analysis model is also subdivided into several "parts" whose number is the same as the number of the Parents, and then each part is subdivided into a number of subdomains. Figure 3 shows an example of hierarchically decomposed mesh for a simplified pressure vessel model. Each Parent stores in its memory a set of the part data, i.e. a number of sets of subdomain data. Each subdomain data include coordinates of nodes, connectivities of elements, material properties, and information on subdomain interfaces. In nonlinear analyses of solid, stresses and displacements are stored in the Parents as well.
We are now developing several kinds of main modules for elastic-plasticity, i.e. ADVENTURE_Solid [11], rigid plasticity, i.e. ADVENTURE_Forge, explicit impact, i.e. ADVENTURE_Impact [14, 15], incompressible fluid flow, ADVENTURE_Fluid [17] and magnetics, i.e. ADVENTURE_Magnetic [13]. Common functions related to finite elements are programmed as class libraries named libFEM.

3.2 Standardization of I/O

Characteristics of I/O data for large scale computational mechanics analyses are summarized as follows :

(1) data size is very huge,
(2) the data must be dealt with in various types of parallel and distributed environments,
(3) the number of data files increases, and
(4) most of the data are of domain-decomposition type.

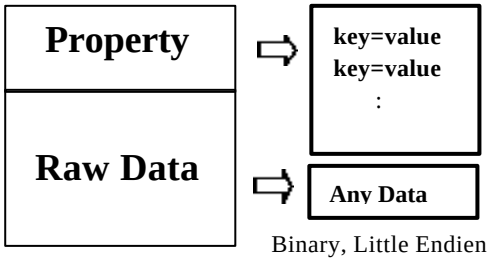

Figure 3 : Media-independent Document.

To efficiently handle such data with characteristics (1) to (3), we have developed "Media Independent Document" as a container of the data [7]. To efficiently deal with domain-decomposition type data for various kinds of mechanics analyses, i.e. characteristic (4), we have also developed a generalized I/O format [7].

The Media Independent Document (simply called the Document in this paper) can exist anywhere in a parallel and distributed environment, that is, memory or files on hard disks of computers connected through a network. As shown in Figure 3, the Document consists of the following two portions : the "Property" portion and the "Mass Data" portion. Stored in the Property portion are control data, the number of items stored in the Mass Data portion, labels and units of the data. The format of the Property portion is rather simple, i.e. key=value, where the value can be randomly accessed using the key. The Mass Data portion stores a huge amount of data such as nodal constraints, and connectivities of elements and analysis results in a binary format. These data can be accessed by several functions prepared in I/O libraries. The byte order of the binary data is unified to Little Endian. Different endian types among different hardware systems are taken care of by the I/O libraries.

Various kinds of input data are required to pass through the domain decomposer. To avoid complex specification for the domain decomposer, depending on different main modules, the model data other than the mesh are generalized in the following. The data used in a finite element analysis consist of a mesh, i.e. nodes and elements, and the data attached on to the mesh such as initial conditions, boundary conditions, material properties and analysis results. The latter data are attached to either nodes or elements. They can be abstracted in terms of the following eight general data types.

fega_type = (All)?(Node|Element)(Variable|Constant)

For example, "NodeVariable" type means that some different sets of data in the same format are attached onto several nodes. The eight kinds of generalized data types are called FEGA (Finite Element Generic Attributes). The FEGA is similar to the UCD (Unstructured Cell Data) format of the commercial visualization software, AVS. The FEGA is stored in the Property portion of the FEGA document as shown in Fig. 4. Sets of data are stored in the Mass Data portion. The domain decomposer of the ADVENTURE system can process any kinds of finite element model data once they are specified with the FEGA.

```
[Properties]
content_type=FEGenericAttribute
num_items=Number of Records
format=<value>'s format [((i|f)Number of Bytes)+]
fega_type=(All)?(Node|Element)(Variable|Constant)
label=Name with Physical Meaning (ex. Young, Poisson)
dd_option=Option for DomainDecomposer
index_byte=Number of Bytes of index (Usually 4(Bytes) because of int32))
(8 byte integer for ultra large scale problem)
[Data]
if fega_type=All(Node|Element)Variable
<value> x num_items
if fega_type=All(Node|Element)Constant
<value> x 1
if fega_type=(Node|Element)Variable
(<value>, <value>)  x num_items
if fega_type=All(Node|Element)Variable
<value>, <value> x num_items
```

Figure 4 : Media-independent document including Finite element generic attributes (FEGA).

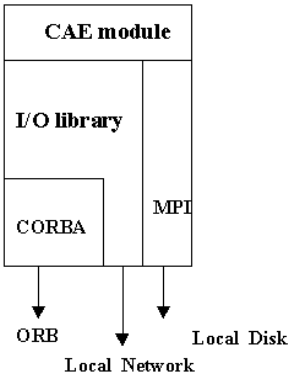

Figure 5(a) : CAE module with ADVENTURE_IO.

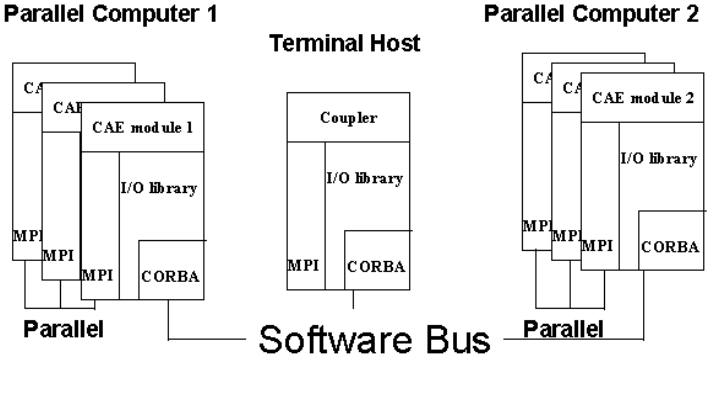

Figure 5(b) : CAE modules collaboration via ADVENTURE_IO.

The ADVENTURE_IO [7], which is the name for the standardized I/O format and libraries, also possesses a generalized collaboration mechanism of multiple modules in a distributed computing environment, employing the CORBA technology. The CORBA is embedded in the ADVENTURE_IO libraries as illustrated in Fig. 5 [17]. By virtue of the mechanism, the modules can directly communicate among others in a prallel and distributed environment in a unified manner, which results in easy collaboration of multiple modules for coupled analyses [17] and optimization calculations.

3.3 Shape Optimization

Owing to the standardization of I/O format and libraries, modules are easily integrated in several ways. Optimization calculations are easily performed by combining the ADVENTURE's pre-, main- and post modules with Optimization modules. In the project, we are developing two kinds of design optimization modules. One is a module for parametric optimization, named the ADVENTURE_Opt, which is designed and programmed based on an object-oriented approach. The module implements various optimization algorithms including mathematical programming, gradient-based methods, simulated annealing, genetic / evolutionary algorithms.
The other is the non-parametric shape optimization module based on the traction method [23], named the ADVENTURE_Shape, which assures the smoothness of boundaries during shape varying processes. With this module, we can solve volume minimization problems of linear elastic continua under constraint on mean compliances with respect to given external forces. Reshaping is accomplished by the traction method in which boundary value problems of partial differential equations are defined, the shape gradient on the boundary is then evaluated and the boundary is varied by acting the traction in proportion to the negative value of the shape gradient.

4 Numerical Examples

This example is a non-parametric shape optimization of support structure of express way as shown in Fig. 6. Figure 7 shows a mesh with one million DOFs. The bottom surface is completely fixed, its right-hand surface was fixed in one direction, and both edges of the I-shape beam were fixed in one direction. Uniform pressure was applied to the top surface of the I-shape beam. In this analysis, a PC cluster consisting 10 PCs (Alpha 21264 (667MHz, 1GB)) was used. Figure 8 shows the obtained optimized shape. For one iteration of the traction force method, the main portion of the traction force method took only 197 seconds, while the portion of the parallel finite element analyses took 77,058

Figure 6 : Support structure of an express way.

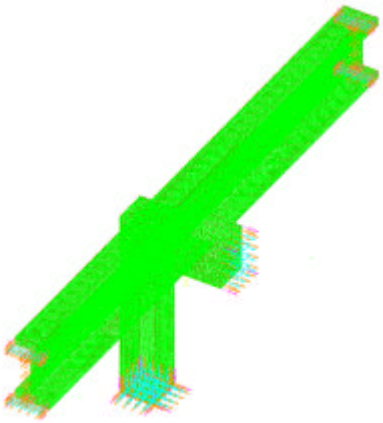

Figure 7 : One million DOF mesh of the half of the support structure.

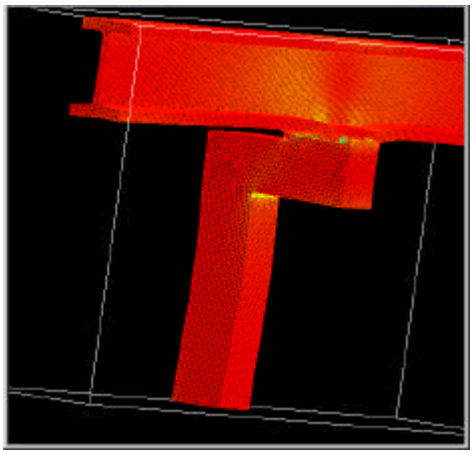

Figure 8 : An optimum shape.

seconds. To get the optimized shape, three iterations for the traction force method were needed. It means that 15 times of the parallel finite element analysis with one million DOF mesh were needed.

In the second example, a parametric optimization of an I-shaped beam with multiple holes was solved using a parallel GA as an optimization algorithm. The goal of the optimization was to minimize the weight of the beam without exceeding the yield stress of material. A PC cluster consisting of 48 Pentium III (500MHz) processors was used. Each parallel stress analysis used 2 processors. The mesh sizes for each analysis varied from around 5,000 to 100,000 elements. Population size of the GA was 48. Each parallel FE analysis took anywhere from 1 to 5 minutes. Figure 9 shows typical finite

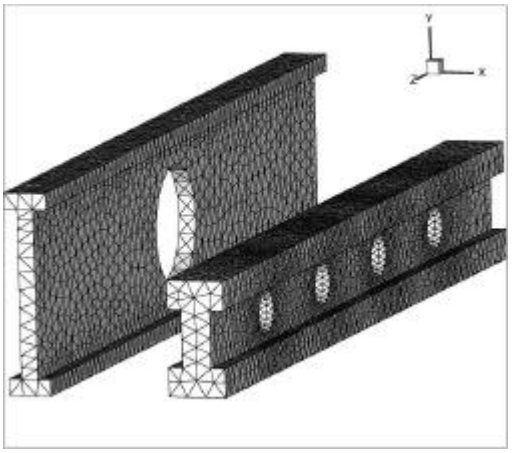

Figure 9 : Typeical meshes.

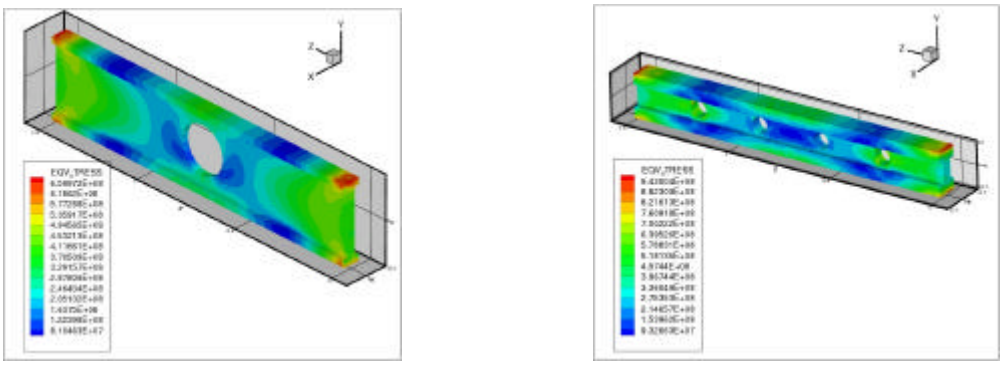

Figure 10 : Stress distributions.

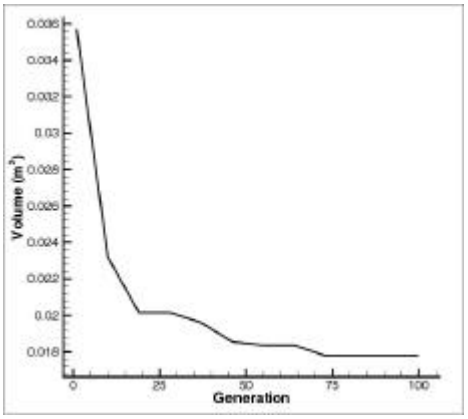

Figure 11 : Convergence of parallel GA.

element meshes, while Fig. 10 shows their stress distributions. Figure 11 shows the convergence history of the beam weight. It took 75 generations to get the optimum solution.

5 Conclusions

The present paper describes the module-based parallel computational mechanics system for large scale analysis and design named ADVENTURE system. After describing the design concept of the system, some key technologies employed to develop the system are presented, i.e. a parallel solver based on the hierarchical domain decomposition, standardization of I/O format and libraries and optimization algorithm. Practical performances of the ADVENTURE system are demonstrated through shape optimization calculations of 3D models.
The first version of the ADVENTURE system has been released from the project website as open source programs since March 1, 2002 [1].

Acknowledgements

This work was performed as a part of Research-for-the Future Program sponsored by the Japan Society for the Promotion of Science. The authors also wish to thank Professor Yagawa, University of Tokyo and all the members of the ADVENTURE project.

References

[1] http://adventure.q.t.u-tokyo.ac.jp/
[2] S. Yoshimura, *Development of computational mechanics system for large scale analysis and design*, Report on Computational Science and Engieering, JSPS-RFTF Program 2001-2002, (2002), pp.61-72.
[3] S. Yoshimura, R. Shioya, H. Noguchi, T. Miyamura, *Advanced general-purpose computational mechanics system for large scale analysis and design*, Journal of Computational and Applied Mathematics, (in Press).
[4] http://proton.is.s.u-tokyo.ac.jp/cse/index-e.html/
[5] http://www.gmd.de/SCAI/europort
[6] S. Tanaka, Y. Wada, S. Yoshimura, G. Yagawa, *A framework for integration of CAE modules in a distributed environment*, Internet Transactions of Japan Society of Computational Engineering and Science, Paper No. 19990026, (1999) (in Japanese).
[7] T. Miyamura, S. Tanaka, H. Takubo, S. Yoshimura, G. Yagawa, *Standardization of Input/Output data in large scale parallel computational mechanics system*, Transactions of Japan Society of Computational Engineering and Science, 2, (2000), 219-226 (in Japanese).
[8] G. Yagawa, S. Yoshimura, K. Nakao, *Automatic mesh generation of complex geometries based on fuzzy knowledge processing and computational geometry*, Integrated Computer-Aided Engineering, 2, (1995), 265-280.
[9] S. Yoshimura, H. Nitta, G. Yagawa, H. Akiba, *Parallel automatic mesh generation of nuclear structures with ten-million nodes*, Transactions of 15th International Conference on Structural Mechanics in Reactor Technology, Seoul, II (1999), pp.21-28.
[10] T. Miyamura, H. Noguchi, R. Shioya, S. Yoshimura, G. Yagawa, *Elastic-plastic analysis of nuclear structures with millions of DOFs using the hierarchical domain decomposition method*, Nuclear Engineering and Design, 212, (2002), 335-355.
[11] R. Shioya, H. Kanayama, D. Tagami, M. Ogino, *3D large scale structural analysis using a balancing domain decomposition method*, Transactions of Japan Society of Computational Engineering and Science, 2, (2000), 139-144 (in Japanese).
[12] R. Shioya, H. Kanayama, D. Tagami, E. Imamura, *A domain decomposition approach for non-steady heat conductive analysis*, Advances in Computational Engineering and Science, 189pdf,

(2001), pp.1-6.
[13] H. Kanayama, R. Shioya, D. Tagami, M. Saito, *Numerical analysis of 3D eddy current problems by the hierarchical domain decomposition method*, Transactions of Japan Society of Computational Engineering and Science, 3, (2001), 151-156 (in Japanese).
[14] A. Oishi, S. Yoshimura, G. Yagawa, *A parallel explicit finite element analysis based on hierarchical domain decomposition*, Transactions of Japan Society of Computational Engineering and Science, 3, (2000), 131-226 (in Japanese).
[15] A. Oishi, S. Yoshimura, G. Yagawa, *Domain decomposition based parallel contact algorithm and its implementation to explicit finite element analysis*, JSME International Journal, 45A, (2002), 123-130.
[16] S. Shoui, S. Yoshimura, H. Akiba, T. Ohyama, G. Yagawa, *Parallel visualization of finite element solutions with ten million DOFs using PC cluster*, Proceedings of European Congress on Computational Methods in Science and Engineering (ECCOMAS2000), Balcelona, CD-ROM, (2000).
[17] Y. Nakabayashi, S. Yoshimura, G. Yagawa, *Fluid-structure weak coupled analysis in parallel and distributed environment*, Computer Methods in Applied Mechanics and Engineering, Submitted for Publication.
[18] G. Karypis, V. Kumar, *Multilevel k-way partitioning scheme for irregular graphs*, Technical Report TR 95-064, Department of Computer Science, University of Minnesota, (1995).
[19] G. Karypis, V. Kumar, *Parallel multilevel k-way partitioning scheme for irregular graphs*, Technical Report TR 96-036, Department of Computer Science, University of Minnesota, (1996).
[20] G. Yagawa, R. Shioya, *Parallel finite elements on a massively parallel computer with domain decomposition*, Computing Systems in Engineering, 4, (1993), 495-503.
[21] C. Farhat, F. X. Roux, Implicit parallel processing in structural mechanics, Computational Mechanics Advances, 2, (1994), 1-124.
[22] J. Mandel, *Balancing domain decomposition*, Communications on Numerical Methods in Engineering, 9, (1993), 233-241.
[23] H. Azegami, Z. C. Wu, *Domain optimization analysis in linear elastic problems (Approach using traction method)*, JSME International Journal, 39A, (1996), 272-278.