Presented at PSE Workshop held in 1st IEEE International Conference on e-Science and Grid Computing, Melbourne, Dec. 5, 2005

A Distributed Problem Solving Environment (PSE) for Scientific Computing

Shigeo Kawata[1)], Hideaki Fuju[1)], Hideaki Sugiura[1)], Yuichi Saitoh[1)], Yoshikazu Hayase[2)], Takayuki Teramoto[3)], Takashi Kikuchi[1)], Hitohide Usami[4)], Hiroyuki Kanazawa[5)], Motohiro Yamada[5)], Yutaka Miyahara[5)], Masahide Fujisaki[5)]

1) Graduate School of Engineering, Utsunomiya University 7-1-2 Yohtoh, Utsunomiya 321-8585, Japan,kwt@cc.utsunomiya-u.ac.jp

2) Department of Electronic Control Engineering, Toyama National College of Maritime Technology, 1-2 Ebie Neriya, Shinminato, Toyama, 933-0293, Japan

3) Department of Electronics and Computer Engineering, Tsuyama National College of Technology Numa 624-1, Tsuyama, 708-8509, Japan

4) National Institute of Informatics, Center for Grid Research and Development, Jinbocho Mitsui-building 14F(NAREGI), 1-105, Kanda-Jinbocho, Chiyoda-ku, Tokyo 101-0051, Japan

5) Computational Science and Engineering Center, Fujitsu Limited, Mihama-ku, Chiba 261-8588, Japan

Abstract

A distributed Problem Solving Environment (PSE) is proposed to help users solve partial differential equation (PDE) based problems in scientific computing. The system inputs a problem description and outputs a program flow, a C-language source code for the problem and also a document for the program. Each module is distributed on distributed computers. The PSE contains all the information of the problem, PDEs, discretization scheme, mesh information, equation manipulation results, designed program structure, variable and constant definitions and program itself. Therefore the documentation support module generates a document for the generated program and the problem itself in the PSE. The module liaison module generates an adapter module among the distributed PSE modules. The job execution service module deploys programs generated or prepared on distributed computer resources and helps users run the programs on the distributed computers. The concept of the distributed PSE extends the potential of conventional PSE systems.

1. Introduction

Recently studies on Problem Solving Environment (PSE)[1-13] for partial differential equation (PDE) based problems have been extensively explored in order to support engineers, scientists, students and so on to compute or simulate their own problems on distributed computers. Computer analyses and simulations have been performed to design products, study scientific issues and more, and have been recognized as the third method following theoretical and experimental methods. One of the major objectives in PSE researches is to help users compute or simulate their problems without heavy tasks, for example, without complete knowledge[10, 11] for computations or without a full programming[1-12]. In this sense the PSE provides an infrastructure for software for computational engineering and science.

In PSE for PDEs, one of problems, which should be solved, is to develop huge PSE systems, including reusability of legacy PSE softwares. In order to solve this problem in PSE, a module-based PSE is proposed[14]; each PSE module solves a part of PSE tasks, for example, problem description interface, discretization, scheme suggestion module, program flow designer, program generator, data analyzer, visualizer, and so on. If each module can be developed independently and works cooperatively and smoothly to solve one PSE job, the heavy work of PSE development may be drastically relaxed.

On the other hand, in High-Performance Computing (HPC) and simulations in science and technology, distributed computer systems including GRID systems [15, 16] are now popular and provide a flexible and cost-effective environment. The distributed computer system may be in a complex environment of heterogeneous computer hardware and software. However, it is quite

difficult for users to know the detail structure of the distributed computer system they use. In a realistic and useful distributed environment users should be supported by assistant software [15], so that users can use the distributed computer systems smoothly and effectively without detail knowledge of the distributed computer system itself.

In this paper a new distributed PSE, called D-NCAS, is proposed in order to support users to generate a computer program and to work on distributed computer systems. The PSE system inputs a problem information including discretization and computation schemes, and outputs a program flow, a C-language source code for the problem and also a document for the program and for the problem. On a host computer a user inputs his/her problem, and the host navigates the user to solve the problem. The distributed PSE for PDEs consists of several modules: a problem description module, a discretization one, an equation manipulation one, a program design one, a program generation one, documentation support module, a module liaison module and a job execution service module. Each module is distributed on distributed computers, and all the information is described by the Extensible Markup Language (XML)[17] including the Mathematical Markup Language (MathML) [18]. Each distributed module communicates with the host module by using XML documents, so that outputs from each module are visualized. Independent modules which are developed by other engineers or users for one of the functions specified above can be also used after adjustments to the distributed PSE interface, if necessary. Therefore the concept of the distributed PSE extends the potential of conventional PSE systems.

The PSE contains all the information of the problem itself, PDEs, discretization scheme, mesh information, equation manipulation results, designed program structure, variable and constant definitions and program itself. Therefore the documentation support module also generates a document for the generated program and the problem itself in the PSE.

The module liaison module generates an adapter module among the distributed PSE modules. The adapter module generated by the module liaison system inputs output data from preceding modules and/or external modules, and connects the data to the input data for the next module.

The job execution service module deploys programs generated or prepared on distributed computer resources and help users run the programs on the distributed computers. The PSE tells users which computers are available and appropriate for their specific application software by using hardware and software information specified. Then the users deploy their software on the distributed computer systems. The PSE may open a new flexible high performance computing environment.

2. Target Issues of PSE

The present research target issues of the PSE include the followings: (1) Computer-assisted program generation should be supported. (2) Simple and easy execution of users' application programs on distributed computers. (3) Simple and easy deploy of users' programs onto the distributed computer environment.

Based on the PSE presented in this paper, users may work smoothly on a distributed computer system: users obtain simulation programs generated by the PSE, also need not have precise information about hardware location behind the PSE, and execute their software without difficulties. The job execution service module in the distributed PSE holds all the information about the distributed computer system and manages all the information relating to hardware, software installed, job execution, network and the location of computation results .

3. A Distributed PSE

In our distributed PSE all the modules are distributed on network-linked computers. The information for the distributed modules and the computers are registered in a host computer. Newly developed modules by some users or scientists or so can be also registered in the host PSE server. The distributed PSE host server has the registered information for the modules oriented to one specific purpose, and users can obtain the information for each module and can select one of the modules to perform one task in all the PSE process. The schematic diagram for the distributed PSE is shown in Fig.1.

At the beginning of problem solving process a user inputs his/her problem information and computation scheme information into the host PSE server. All the information is described by XML, and is visualized to the user.

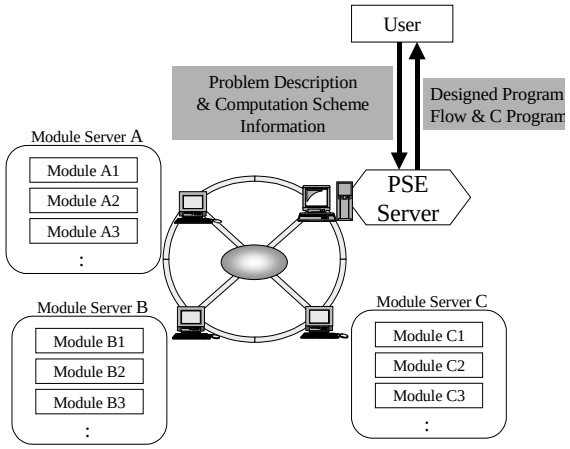

Fig. 1. Overview of the distributed PSE

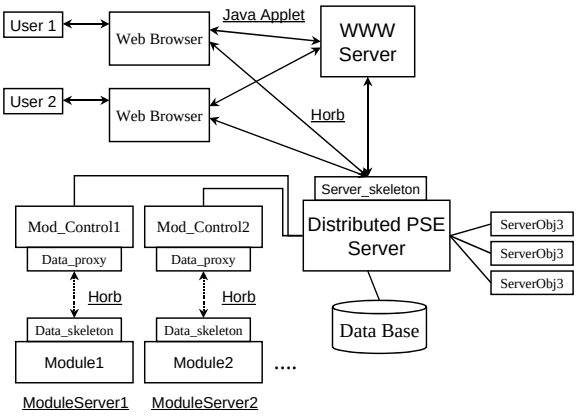

Fig. 2. Structure of distributed PSE Network

The communication is accomplished through an interface using WWW server and Applet. The PSE server sends an information described by XML to a module, and the module performs the task. The module sends the result based on the input XML information back to the PSE server. The result is visualized so that the user can check if the result is appropriate. After successive processes, finally the PSE generates a designed program flow and then a C program. Figure 2 shows the network structure employed in the PSE. The program generation PSE module provides a workflow shown in Fig.3, and the user follows the workflow navigation for a problem generation.

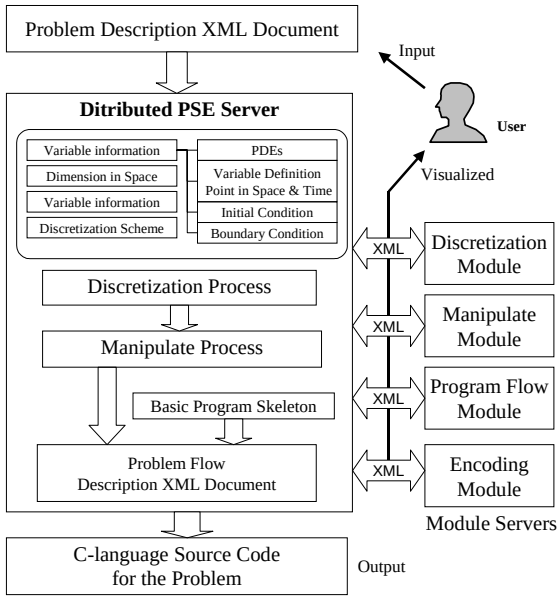

Fig. 3. Distributed-PSE work flow

4. Problem Solving in Distributed PSE

4.1 PSE modules for program generation support

In a problem description XML document the followings are specified: dimension in space, PDEs, constants, variables, variable definition points in space and time, initial and boundary conditions, discretization scheme, and space mesh definition. Because all the information is described by XML, the information is treated as tree-type information. Figure 4 shows a part of an example input problem description in XML, and the XML document is visualized to the user as shown in Fig.5. The PDEs and boundary conditions are also described by the Math-ML (a subset of XML) as shown in Fig. 6. In the XML document for problem description, a user inputs information for dependent and independent variables, constant factors, in addition to the definition in a space mesh and a time mesh for variables. Based on the information with the mesh information and the discretization scheme information, the PDEs are discretized and manipulated in order to obtain a matrix or a final form of each discretized PDEs. First we present example results of PDEs discretization by finite difference method (FDM).

The input problem information includes PDEs and a discretization scheme specified by a user. An FDM/FEM module transforms a PDEs XML document to another XML document for discretized PDEs, based on the input information.

An Example discretization is presented in the followings: an example PDE for diffusion problem is Eq.1.

$$\frac{\partial f}{\partial t} + A\frac{\partial^2 T}{\partial x^2} + A\frac{\partial^2 T}{\partial y^2} = 0 \qquad (1)$$

When a user chooses the Euler explicit scheme, the first term is discretized as follows:

```
<?xml version="1.0" encoding="SHIFT_JIS"?>
<?xml-stylesheet type="text/xsl" href="fdmXSL.xsl"?>

<informationFDM>
  <header>
    <title>Thermal Diffusion Problem</title>
    <dimension>2</dimension>
  </header>
  <mesh_condition>
    <time>
      <factor>t</factor>
        <time_type>uniform</time_type>
          <times sec="100"/>
          <timestep sec="0.01"/>
    </time>
    <space>
      <space_type>uniform</space_type>
        <i_coordinate imin="0" imax="20"
istep="0.1">
          x</i_coordinate>
        <j_coordinate jmin="0" jmax="10"
jstep="0.1">
```

Fig. 4. An example problem description XML document

$$\left.\frac{\partial f}{\partial t}\right|_i = \frac{f^{n+1}{}_i - f^{n}{}_i}{dt} \qquad (2)$$

where the superscript of '*n*' is a time index , the subscript of '*i*' is space index and '*dt*' shows the time-step size. The second and the third terms are discretized as shown in Eq. 3, depending on the user's specification. In this case the central difference scheme is employed.

$$\left.\frac{\partial^2 T}{\partial x^2}\right|_i = \frac{T^{n}{}_{i+1} - 2T^{n}{}_i + T^{n}{}_{i-1}}{dx^2} \qquad (3)$$

where '*dx*' is the mesh size. Figure 6 shows a problem information described by an XML document for this example. The following example shows an advection equation, where the advection term is discretized by the Upwind difference scheme.

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} = 0 \qquad (4)$$

where '*u*' is a velocity of a convective material, for example, fluid. In this case the second term in Eq.4 may be transformed to Eq.5

$$\left.u\frac{\partial f}{\partial x}\right|_i = \frac{u^{n}{}_i + |u^{n}{}_i|}{2}\frac{f^{n}{}_i - f^{n}{}_{i-1}}{dx} + \frac{u^{n}{}_i - |u^{n}{}_i|}{2}\frac{f^{n}{}_{i+1} - f^{n}{}_i}{dx} \qquad (5)$$

Figure 7(a) shows the variable definition points of '*f* ' and '*T* ' in Eq. 1 on a mesh. Figure 7(b) shows an visualization example for Eq.1 using the XML document. The XML and Math-ML documents are visualized in a web browser by using the software "techexplorer"[19]. The discretization result is also shown in Figure 7(c).

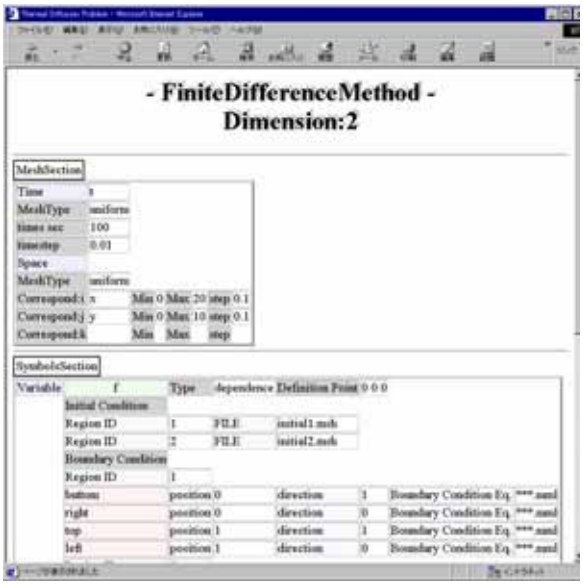

Fig. 5. Visualization of an XML Problem Information using an XSL Style Sheet

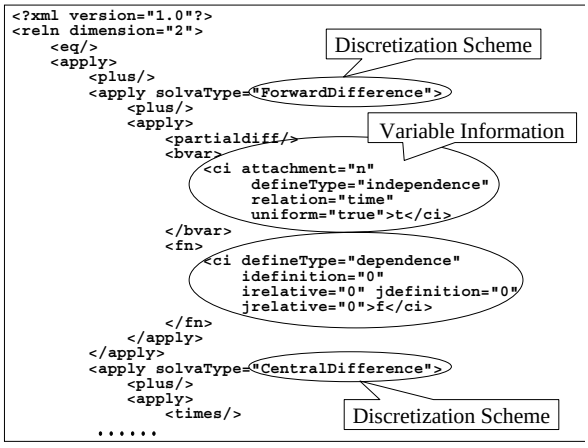

Fig. 6. A Problem Information for FDM in XML

The PSE problem input module sends an input information described in XML to the FDM/FEM module, and the module performs the discretization and equation manipulation tasks. Then the FDM/FEM module sends the result written in the XML document back to the PSE host.

The program-generation PSE modules also help users generate MPI-based parallel simulation programs based on partial-differential equations (PDEs). The D-NCAS capability explores possibilities to visualize and steer the parallel program design process. At present D-NCAS supports a domain decomposition in a design of a parallel numerical simulation program, and the domain decomposition is designed or steered by users through a visualization window. After designing the domain decomposition, the parallel program itself is also designed

and generated in NCAS, and the designed parallel program is visualized and steered by a PAD diagram.

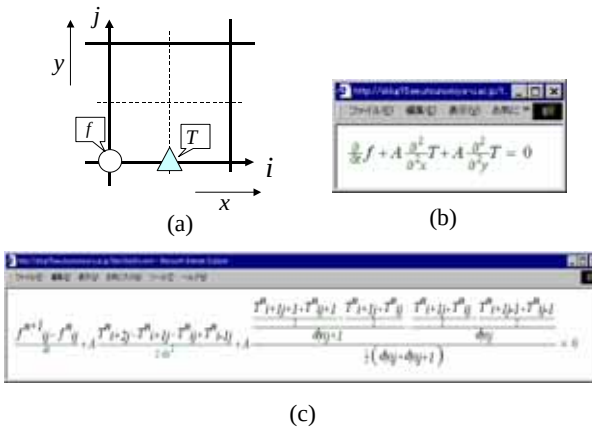

Fig. 7. a) Variable definition positions in one mesh, b) a visualized example basic equation and c) a discretization result.

In D-NCAS, MPI functions are employed for message passing, and a SPMD (single program multiple data) model is supported. The visualization and steering capabilities provide users a flexible design possibility of parallel programming.

The parallel program generation module of D-NCAS inputs PDEs, boundary and initial conditions, parallel program information for a domain decomposition, CPU number employed, and mesh information. All the processes are visualized and steered in order to meet the user's requirement. Figure 8 shows an example of the visualization for a generated program by a PAD diagram.

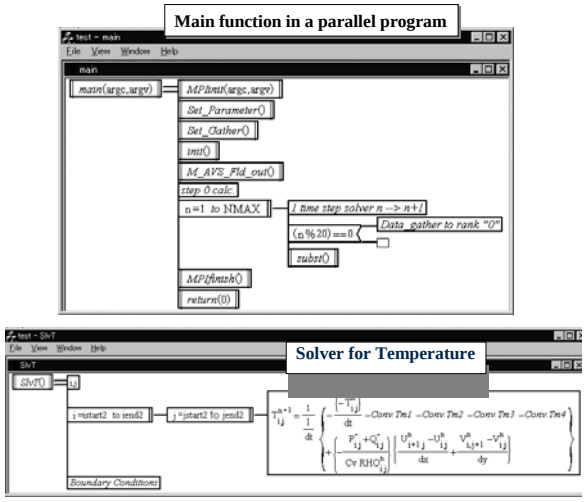

Fig. 8. Example visualization of a main part of a designed parallel program by using a PAD diagram in the parallel program generation component in the PSE toolkit.

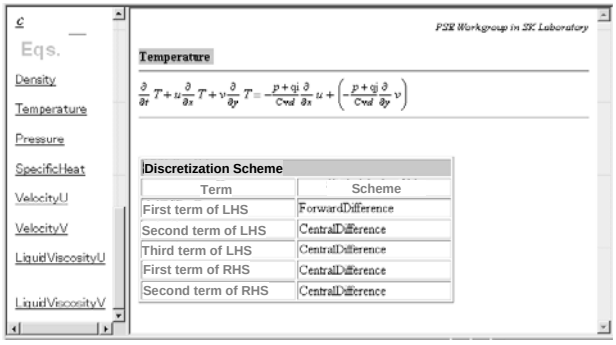

Fig. 9. Example of a document generated by the parallel program generation component in the PSE server. This module contains the input information of abstract problem input information, for example, PDEs, problem area, variables, etc. Therefore the document is also generated with the MPI-based parallel program from the higher level information.

At the same time the program generation module also generates a document for the generated program as shown in Fig. 9, because the D-NCAS component keeps all the the information from the abstract problem description, equations and so on as described above.

4.2 Module Liaison for a distributed PSE

In our distributed PSE (D-NCAS) all functions from an input problem description to a C program generation and a document generation are divided into separate functions. They are assigned to specific modules distributed on network-linked computers. The

information for the distributed modules and the computers is registered in a host computer, that is, a module register PSE server. New modules developed by users can also be registered in the host module register PSE server. Users input their problem information into the PSE server through the PSE WEB server. The PSE server sends the information described in XML to the next module, and the module performs the task. The module sends the result based on the input XML information back to the PSE server. After successive processes, finally the PSE server gets a designed program flow and then a C program. Figure 10 shows the structure of the module liaison module.

Each module of the distributed module-based PSE may be developed by different users. Although this feature makes the PSE flexible and expandable as discussed in Introduction, the input/output data type and order may be different from those of other modules. If they do not fit to those for other modules, the problem solving process can not be accomplished.

Our module liaison system provides the data adapter to convert the input/output data to the data fitted to those for the next module. As long as the input/output data are described by XML and their information including the data type and the data order are known, the module liaison system produces a data adapter to connect the modules in the distributed PSE. In order to know the input/output data information, in this paper we assume that each module has an example set of the input/output data information described by XML, besides a complete document for the data input/output. Without the full information of the input/output data one can not use the module in a problem solving process. Therefore this assumption is reasonable.

Figure 11 shows the interface of the module register PSE server. The module register PSE server contains the module information, including the input/output data information, the location of the document and the module location URL (Uniform Resource Locator), and it also has the information of user-designed workflows to solve a problem. Through the WEB page shown in Fig.11 users find the available module list. The users connect the selected modules, and design the workflow for their problem solving tasks.

At the module registration phase, the module information of the module class name, its method name and arguments are registered together with the data input/output information, the module location, etc. The module information is described by WSDL (web service description language). When the module is registered to the module register PSE server and is deployed on distributed computers, the WSDD (web service deployment descriptor) file is also used as a usual web service.

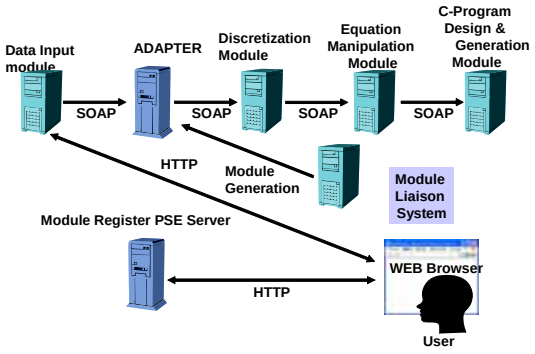

Fig. 10. A module liaison system

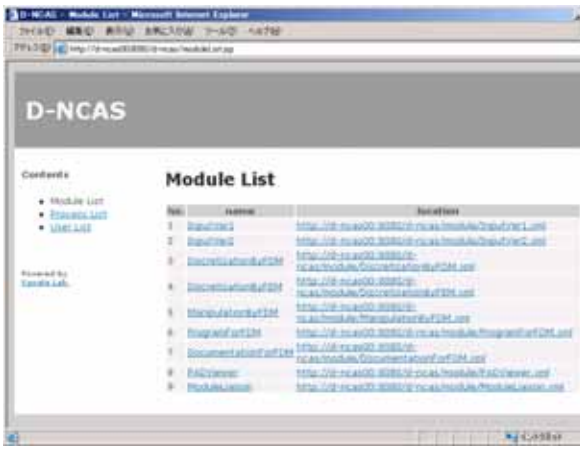

Fig. 11. Interface of the module register PSE server

In a distributed PSE a new module may be added or a legacy module may be updated by individual users/developers. If the module liaison system helps users connect the modules and provides adaptors among the modules based on the input/output data information of the modules, the module liaison system elevates the module-based distributed PSE capability further.

Figure 12 shows a concept of the data adapter module, generated by the module liaison system. The preceding modules output data and the next module may input the data together with an external data in general. Each module has the input/output data information including example input/output data XML files and a document for the data.

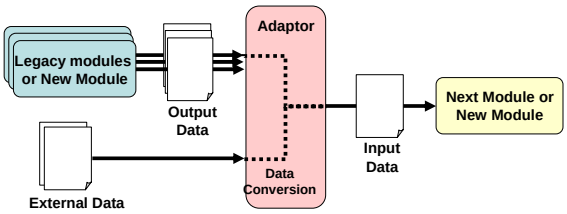

Fig. 12. Adapter generated by the module liaison system. The adapter converts the output data from preceding modules and may combine the external data to produce the input data to the next module.

4.3 Job execution service module PSE

The scientific computing tends to require high performance computers, which may be distributed locally or globally. At present network linked distributed computer systems including Grid systems are widely available for researchers and engineers. However, it is difficult for researchers to obtain detail information on distributed hardware and software resources, which may be a large system, consisted of many computers. When the users work on the distributed resources, the users need to find computing resources, data analysis servers and data storage servers. The job execution management including these functions is essentially important for scientific computing on the distributed resources.

The job execution service module does not need any special Grid middleware to construct it, and the client can use this system by accessing the Web page for the job execution service system.

The job execution service module consists of dynamic system management servers, execution servers and data servers. The dynamic system management server is duplicated in order to keep the system robust, and has an assistant management server. The dynamic system management server has a function of the job execution system management, including software deployment, program compilation, job execution, job status retrieval and computing data retrieval. This system does not require special middleware such as Globus or UNICORE or GLite or etc. Users access the web page on the dynamic system management server, and the clients submit jobs. After the submitted job finishes, the dynamic system management server collects the information from other distributed computers. The dynamic management server and its assistant server move dynamically to new servers, if the present servers become busy. The dynamic system management server also demands the execution server to transfer the result data to the optimal data server. The dynamic system management server copies the computing data and sends the compressed computing data to another optimal data server in order for a robust data storage system. The clients can deploy their programs, execute jobs and retrieve the result data by accessing only the web page in the dynamic system management server. This job execution management server also has a function of automatic system construction, so that the users can manage the setup of the job execution management system easily on their closed distributed computers.

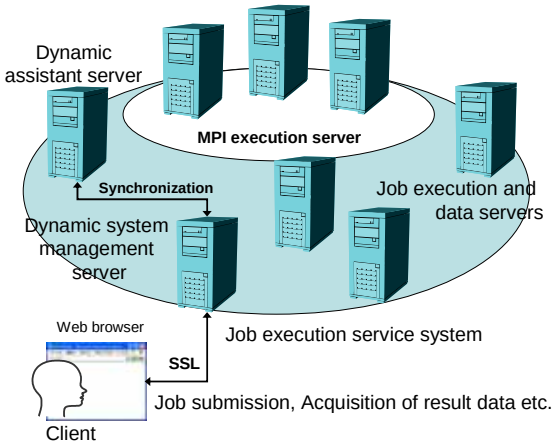

Fig. 13. Structure of the job execution service system

5. Conclusions

In this paper we presented the distributed PSE of D-NCAS to help users work on distributed computers. The role and viability of the PSE in the distributed computer system are demonstrated. The PSE server provides a smooth and flexible environment in the HPC on the distributed computers. The distributed PSE supports generation of simulation program and its documentation for PDEs-based problems, and also supports the job execution task on a distributed computer environment. The job execution service module in the distributed PSE encapsulates the complex information of distributed system so that on the PSE users can perform HPC as if distributed computers are under users' hand. The distributed PSE may open a new environment for HPC world.

Acknowledgements

This work was partly supported by the NAREGI (National Research Grid initiative) project in Japan. The authors would like to express their appreciations to Prof. K. Miura (the leader of the NAREGI project), Dr. Motohiro Yamada, Dr. Yutaka Miyahara, Prof. yoshio Tago, Prof. Yukio Umetani and Dr. Hiroyuki Kanazawa for their fruitful discussions on this subject.

References

[1] C.Boonmee and S.Kawata, “Computer-Assisted Simulation Environment for Partial-Differential-Equation Problem, 1. Data Structure and Steering of Problem Solving Process”, *Trans. of the Japan Society for Computational Engineering and Science*, Paper No. 19980001, 1998.

[2] C.Boonmee and S.Kawata, “Computer-Assisted Simulation Environment for Partial-Differential-Equation Problem: 2. Visualization and Steering of Problem Solving Process”, *Trans. of the Japan Society for Computational Engineering and Science*, Paper No. 19980002, 1998.

[3] S.Kawata, C.Boonmee, A.Fujita, T.Nakamura, T.Teramoto, Y.Hayase, Y.Manabe, Y.Tago and M.Matsumoto, “Visual Steering of the Simulation Process in a Scientific Numerical Simulation Environment –NCAS- ", *Enabling Technologies for Computational Science*, edited by E.Houstis and J.Rice, Kluwer Academic Pub., 2000, pp. 291-300.

[4] E.Gallopoulos, E.Houstis, and J.R.Rice, “Future Research Directions in Problem Solving Environments for Computational Science”, Technical Report CSRD Report No.1259, *Report of a workshop on Research Direction in Integrating Numerical Analysis, Symbolic Computer, Computational Geometry, and Artificial Intelligence for Computational Science*, Washington DC., April 11-12, 1991.

[5] Y.Umetani, "DEQSOL A numerical Simulation Language for Vector/Parallel Processors", Proc. *IFIP TC2/WG22*, 5, 1985, pp. 147-164.

[6] Y.Hirayama, J.Ishida, T.Ota, M.Igai, S.Kubo, S.Yamaga, "Physical Simulation using Numerical Simulation Tool PSILAB", *The 1st Problem Solving Environment Workshop*, pp. 1-7, 1998.

[7] J.R.Rice and R.F.Boisvert, "Springer Series in Computational Mathematics 2", *Solving Elliptic Problems Using ELLPACK, Springer-Verlag*, New York, 1984.

[8] H.Fujio and S.Doi, "Finite Element Description System,as a Mid-Layer of PSE", *Proceedings of Conference on Computation Engineering and Science*, Vol.3, No. 2, 1998, pp. 441-444.

[9] T.Okochi, C.Konno, and M.Igai, "High Level Numerical Simulation Language DEQSOL for Parallel Computers", *Trans. of Information Processing Society of Japan*, Vol.35, No.6, 1994, pp. 977-985.

[10] E.N.Houstis, and J.R.Rice, "Parallel ELLPACK, a development environment and problem solving environmentfor high performance computing machienes", edited by In P.Gaffney and E.N.Houstis, *Programming Environments for Hihg-Level Scientific Problem Solving*, North-Holland, Amsterdam, 1992, pp. 229-241.

[11] M.Kubota, T.Kawai, "Automatic Distributed System Simulator Using Nature’s Algorithm on Highly Parallel Computers", *Int. Conf. On Supercomputing in Nuclear Applications*, Mito, March 1990.

[12] A.Fujita, T.Teramoto, T.Nakamura, C.Boonmee, S.Kawata, "Computer-Assisted Parallel Program Generation System P-NCAS from Mathematical Model-Visualization and Steering of Parallel Program Generation Process-", *Trans. of the Japan Society for Computational Engineering and Science*, Paper No. 20000037, 2000.

[13] K.Hoshi, "Development of A Problem Solving Environment for Science and Engineering", *The 1st Problem Solving Environment Workshop*, 1998, pp.13-18

[14] S.Kawata, A.Fujita, S.Machide, T.Hirama, K.Yoshimoto, "Computer-Assisted Simulation System NCAS and a Future PSE", *Proc. the 2nd Problem Solving Environment (PSE) Workshop*, 1999, pp.89-94,

[15]Global Grid Forum. http://www.gridforum.grg/

[16] T. Teramoto, T. Nakamura, S. Kawata, S. Matide, K. Hayasaka, H. Nonaka, E. Sasaki and Y. Sanada, “A Distributed Problem Solving Environment (PSE) for Partial Differential Equation Based Problems”, Trans. Jpn. Soc. Comp. Sci. Eng., Paper No. 20010018, 2001. Available: http://save.k.u-tokyo.ac.jp/jsces/trans/trans2001/No20010018.pdf

[17]W3C Extensible Markup Language. http://www.w3.org/XML/

[18]W3C Math Home. http://www.w3.org/Math/

[19]techexplorer, "IBM techexplorer Hypermedia Browser", http://www.ibm.com/software/network/techexplorer/