Web based metacomputing using DEDIP: application development perspectives

A. K. Aggarwal*, S. C. Patodi**, Paresh Patel§, and Haresh S. Bhatt¥

* Professor & Head
Department of Computer Science
Rollwala Computer Centre
Gujarat University
Ahmedabad – 380 009
Email: aka19@hotmail.com

** Professor
Faculty of Technology & Eng.
M. S. University of Baroda
Baroda - 395001
Email: appmech@vsnl.com

§ Lecturer
Dept. of Civil Eng.
Nirma Institute of Technology
Ahmedabad – 382 481
Email: parpat1@yahoo.com

¥ Scientist "SE"
ND/ITAG/SITAA
Space Application Centre (ISRO)
Ahmedabad – 380 053
Email: haresh@ipdpg.gov.in

Abstract

Development of a parallel application, a distributed application or a web-based application is quite a challenging job. DEDIP is a metacomputing environment which makes such a development very easy. DEDIP was mainly developed for image processing applications. However, users from other fields can use it with advantage. This paper presents the use of DEDIP in two such fields, one in civil engineering for a parallel processing application development and the second in a web-based application developed for distributed processing over Internet.

1. Introduction

Network of workstations has been used for parallel and distributed processing for years. PVM [1] was considered as the parallel machine of a poor man. The invention of Internet has increased the scope of usage of network of workstation.

Internet was invented mainly for information sharing. However it can also be used for parallel processing and distributed processing. Meta computing is the emerging technology that uses the Internet for parallel and distributed processing. Many scientists are working in this field for many years and different metacomputing environments are being developed [2-11]. DEDIP is one of the metacomputing environment that was developed in a research project at Gujarat University mainly for parallel and distributed image processing [10,11]. It has been used operationally at the Space Applications Centre, Ahmedabad.

This paper presents the usage of DEDIP in parallel application & web-based application development in the fields other than the image processing. We have considered one civil engineering application for parallel processing and a group of web-based applications for distributed processing.

2. Metacomputing Environment

Many scientists are working in the field of metacomputing [2-11] providing different facilities to the users.

JavaParty [2] provides mechanisms for transparently distributing remote objects. ParaWeb [3] is an implementation of the JVM that allows Java threads to be transparently executed remotely. Charlotte [4] provides a high level solution that decouples programming environment from the execution. Its disadvantage is that the programmer does not have explicit control over resource utilization. However, its eager scheduling enables the runtime systems to efficiently provide load balancing. Popcorn [5] provides a Java API for writing parallel programs for Internet distribution. Applications are decomposed, by the programmer, into small, self-contained subcomputations, called computelets. The Popcorn is based on buyer-seller concept. It has a centralized entity called the market that determines which CPU seller executes the computelet. Javelin [6] is an infrastructure for Internet based parallel computing. Any free computer system can volunteer to execute a task using the applets supported by Javelin. It follows a client-broker-server architecture. Bayanihan [7] and Ninflet [8] are also very similar to the Javelin.

The methods, reported in most of the above, concentrate in providing computation power to a large and complex application efficiently. All of them expect efficient parallel and distributed programming skills. Their definition of ease of use is around application compilation, scalability, load balancing, fault tolerance, etc.

The WebFlow [9] provides Java-Swing based visual programming environment for metacomputing using Java.

The programmer needs to use Java for metacomputing in the above models. Furthermore, the GUIs of the above models support the monitoring and controlling of an application in the stand-alone mode. Therefore, they do not require elegant & easy GUI for simultaneous execution, monitoring and controlling of multiple applications.

DEDIP [10,11] concentrated on the vast community of scientific users rather than on the efficient programmers for parallel and distributed computing. It made the distributed application development very easy. It supports all languages like Fortran, C, C++ and Java. Its GUI supports all the needs of operational environment executing multiple heterogeneous applications simultaneously. It has its own backend support for process scheduling and monitoring.

3. DEDIP Overview

DEDIP was mainly developed for helping the image processing scientist for parallel and distributed processing over a network of heterogeneous system consisting of VAX/VMS and Unix work stations. Later it was extended for other platforms using Java. It is using Internet for parallel and distributed processing. The DEDIP makes the application development very easy. The user has to simply visualize the parallel & distributed processing possibilities in his application. He then has to divide the application in small tasks accordingly. He should interface all tasks using intermediate files and test the complete applications on a single machine. DEDIP provides browser based GUI to the user

to configure his tasks to make a parallel or distributed application as per his visualization. DEDIP builds the application components (tasks) on remote machine as per the user configuration. DEDIP provides a full-fledged operation environment for executing the user application. The entire DEDIP user interface is user friendly. It is browser based, enabling the user to operate from any node over the Internet.

4. DEDIP applications

DEDIP has proved its usage in the field of image processing. We have used DEDIP in two other fields. One is in civil engineering for parallel processing using a network of Windows & Unix work stations. The second is for developing a group of web based applications.

4.1 Distributed parallel processing

Under a research project work, being carried out at Nirma Institute of Technology (affiliated to Gujarat University) jointly with M.S. University for solving structural analysis problem in the field of civil engineering. We have used DEDIP, instead of PVM or any other parallel machine, to carry out the feasibility study.

Analysis of large-sized complex structure such as multistoried buildings, long span bridges and tall towers involves formulation and solution of a large number of equations. Either matrix stiffness method or finite element method can be used to solve such complex structures. The analysis of such large structures, involving thousands of unknowns, may be expedited by subdividing it into smaller parts referred to as substructures.

In the substructure technique [12], each substructure is analyzed separately and the results are combined to yield the displacements and stresses in the actual structure. The use of substructure technique depending on the size of each substructure and the number of substructures, may results in a considerable saving of storage (about 30 to 40%) and nearly 10 to 20% saving in computational time. To speed up the analysis time, the parallel processing approach can be used. A hardware system, dedicated to parallel processing, is expensive. Parallel processing over a network of workstations is an economical alternative. A network of workstations can be visualized into a distributed computing environment for tackling each substructure on a separate workstation using message passing functions [13] for the communication of data between the computers.

4.1.1. The Sub Structural Technique

The method of sub-structuring for static structural analysis is based on subdividing the large structure into smaller parts, which is known as substructure to obtain the relationship between forces and displacements at the common interfaces or boundaries. These boundary variables are then determined and are used to obtain the unknowns within each substructure. The division of the structure into smaller parts is

totally left to analyst, but it will affect the communications between the computer and subsequently the efficiency of computation.

In the displacement formulation for structural analysis, the basic equation is the equilibrium equation applied to the structure as a whole and is given by

$$[K]\ \{r\} = \{P\} \qquad \text{.....(1)}$$

Where [K] is the stiffness matrix, {r} is the displacement vector and {P} is the load vector.

Using the substructure technique, the above equilibrium equation is obtained by the assemblage of substructure equations. For each substructure, the stiffness matrix, the displacement vector and the load vector are partitioned corresponding to internal and boundary degrees of freedom $\{d_i\}$ and $\{d_b\}$ respectively as follows:

$$\left[\begin{array}{c|c} [k_{ii}] & [k_{ib}] \\ \hline [k_{bi}] & [k_{bb}] \end{array}\right] \quad \left\{\begin{array}{c} \{d_i\} \\ \hline \{d_b\} \end{array}\right\} = \left\{\begin{array}{c} \{Q_i\} \\ \hline \{Q_b\} \end{array}\right\} \qquad \text{.....(2)}$$

In the above equation, a boundary node is defined as a node which is a part of more than one substructure and the degrees of freedom at the boundary nodes are termed as boundary degrees of freedom.

Now the analysis can be performed in two stages,

1. Considering degrees of freedom at the boundaries as fixed, each substructure is analysed on different computers in parallel. Denote the solution obtained from this step by a superscript α.
2. Combine the condensed stiffness of the substructures from different computers to get the global structure stiffness matrix and analyse the assemblage by releasing the boundary degrees of freedom. Denote the processing carried out in this step by a superscript β.

The displacement and load vectors can now be expressed as the sum of above two cases as,

$$\left\{\begin{array}{c} \{d_i\} \\ \hline \{d_b\} \end{array}\right\} = \left\{\begin{array}{c} \{d_i^{\alpha}\} \\ \hline \{d_b^{\alpha}\} \end{array}\right\} + \left\{\begin{array}{c} \{d_i^{\beta}\} \\ \hline \{d_b^{\beta}\} \end{array}\right\} \qquad \text{.....(3)}$$

and

$$\left\{\begin{array}{c} \{Q_i\} \\ \hline \{Q_b\} \end{array}\right\} = \left\{\begin{array}{c} \{Q_i^{\alpha}\} \\ \hline \{Q_b^{\alpha}\} \end{array}\right\} + \left\{\begin{array}{c} \{Q_i^{\beta}\} \\ \hline \{Q_b^{\beta}\} \end{array}\right\} \qquad \text{.....(4)}$$

where subscript i and b denote the terms corresponding to the internal and boundary degrees of freedom respectively.

Obviously as $\{d_b^{\alpha}\}$ is the displacement at the boundary degrees of freedom, when boundaries are fixed, it will be zero. Thus

$$\{d_b^{\alpha}\} = \{0\} \qquad \text{…….(5)}$$

Also in the first stage of the analysis, all the forces are applied at the internal nodes of the substructure and hence these forces do not appear at the second stage. Hence,

$$\{Qi^{\beta}\} = \{0\} \text{ and } \{Qi^{\alpha}\} = \{Qi\} \qquad \text{…….(6)}$$

STAGE I : ANALYSIS WITH FIXED BOUNDARIES : Substituting the value of $\{d_b^{\alpha}\} = \{0\}$ from Eq. 5 into the equilibrium Eq. 2, the set of equations for the first stage of analysis with boundaries of substructure fixed can be written as,

$$\left(\begin{array}{c|c} [k_{ii}] & [k_{ib}] \\ \hline [k_{bi}] & [k_{bb}] \end{array}\right) \quad \left\{\begin{array}{c} \{d_i^{\alpha}\} \\ \hline \{0\} \end{array}\right\} = \left\{\begin{array}{c} \{Q_i\} \\ \hline \{Q_b^{\alpha}\} \end{array}\right\} \text{…….(7)}$$

Solving the first set of above equation ,

$$\{d_i^{\alpha}\} = [k_{ii}]^{-1} \{Q_i\} \qquad \text{……..…(8)}$$

Substituting the value of $\{di^{\alpha}\}$ in the second equation

$$\{Q_b^{\alpha}\} = [k_{bi}][k_{ii}]^{-1} \{Q_i\} \qquad \text{……(9)}$$

Here $\{Q_b^{\alpha}\}$ is the force required to be applied at the substructure boundaries to keep the boundary displacements equal to zero. The above analysis is performed on all the substructures in parallel on different computers .

STAGE II : ANALYSIS WITH BOUNDARIES RELEASED : Again substituting the value of $\{Q_b^{\beta}\}$ in Eq. 2 the set of equations for the second stage of analysis with boundaries released can be written as,

$$\left(\begin{array}{c|c} [k_{ii}] & [k_{ib}] \\ \hline [k_{bi}] & [k_{bb}] \end{array}\right) \quad \left\{\begin{array}{c} \{d_i^{\beta}\} \\ \hline \{d_b^{\beta}\} \end{array}\right\} = \left\{\begin{array}{c} \{0\} \\ \hline \{Q_b^{\beta}\} \end{array}\right\} \text{…….(10)}$$

Solving the first set of equation we have,

$$\{d_i^{\beta}\} = -[k_{ii}]^{-1}\,[k_{ib}]\{d_b^{\beta}\} \qquad \ldots\ldots\ldots\ldots(11)$$

Solving the second set of equation we get,

$$[k_{bi}]\{d_i^{\beta}\} + [k_{bb}]\{d_b^{\beta}\} = \{Q_b^{\beta}\} \qquad \ldots\ldots..(!2)$$

Substituting from Eq. 11 for $\{d_i^{\beta}\}$ into Eq. 12 we get,

$$-[k_{bi}][k_{ii}]^{-1}\,[k_{ib}]\{d_b^{\beta}\} + [k_{bb}]\{d_b^{\beta}\} = \{Q_b^{\beta}\} \qquad \ldots\ldots\ldots.(13)$$

or

$$[k^*]\ \{d_b^{\beta}\} = \{Q_b^{\beta}\} \qquad \ldots\ldots\ldots\ldots...(14)$$

where

$$[k^*] = [k_{bb}] - [k_{bi}][k_{ii}]^{-1}\,[k_{ib}] \qquad \ldots\ldots\ldots..\ldots(15)$$

The Eq. 15 is the equilibrium equation for the substructure in terms of its boundary degrees of freedom and [k*] is the corresponding stiffness matrix called as condensed stiffness matrix. This analysis will be carried out in parallel for all the substructures on different computers and the condensed stiffness matrix for each substructure are assembled to form the global structure stiffness matrix. Thus,

$$[K] = \sum_{s=1}^{s=n} [k^*]s \qquad \ldots\ldots\ldots\ldots\ldots\ldots.(!6)$$

and

$$\{P\} = \{Q_b\} - \sum_{s=1}^{s=n}\{Q_b^{\alpha}\}s \qquad \ldots\ldots\ldots\ldots..\ldots(17)$$

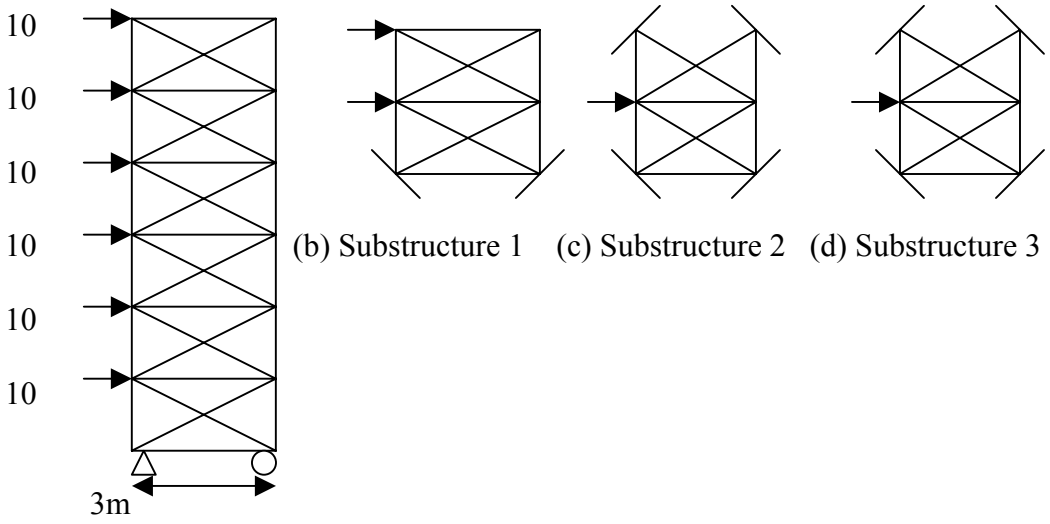
In the above equations n stands for the number of substructures which is equal to the number of computers. The assemblage of the substructures through Eq. 16 and 17 leads to Eq. 1 where all the degrees of freedom are along the common boundaries of the substructures. Solution of Eq. 1 gives the global displacements along the boundaries of the substructure. Now, picking up the appropriate displacements, the vector$\{d_b^{\beta}\}$ can be obtained for each substructure, which will be communicated to different computers and from that $\{d_i^{\beta}\}$ can be determined as per Eq. 11.

Thus all the values of {d} required in Eq. 3 are known for each substructure and from that other quantities like member end forces, stresses and strains can be calculated.

4.1.2. Implementation method

Here the method of substructure technique is illustrated with the help of a simple example of a plane truss by subdividing into three substructures. Obviously, it is easier to analyze this structure without subdividing it into substructures. Therefore

its analysis by the method of substructures using distributed computing strategy is presented here solely for the purpose of demonstration of use of DEDIP environment in structural engineering.



Complete Structure

FIG 1 : EXAMPLE OF ANALYSIS OF A PLANE FRAME

Fig. 1 (a) show the whole truss of width 3m and height 18m subjected to horizontal load. The number of unknowns in complete structure is 25. The whole structure is divided in to three substructures as shown in Fig. 1 (b), (c) and (d). Using substructure technique, the number of unknowns for substructure 1 is 8, for substructure 2 it is 4, and for substructure 3 it is 4. These structures are solved in parallel on three computers and condensed stiffness matrix is assembled for the whole structure, where the number of unknown is 9. So the number of equations to be solved at any time will be reduced, which will increase the computational efficiency.

4.1.3. Parallel processing using DEDIP

Such an implementation using either parallel machine or PVM needs dedicated efforts. Civil engineers need to develop the skills in parallel programming that includes the process forking, joining, synchronization, data communication issues, etc. Furthermore he needs to learn the special debugging techniques for testing his parallel application. DEDIP helps him in avoiding all such additional overheads.

Structural analysis application was divided into 5 small tasks named Parant1, Parant2, ..Parant5. The interface among different tasks is carried out using intermediate files. For example, A1.dat is used between Parant1 and Parant2. These tasks were executed on a single system in accordance with the required sequence to carry out the functional test.

All the tasks are then inter linked using DEDIP GUI to configure the application for parallel processing. Figure 2 shows screen shot depicting the required interdependency.

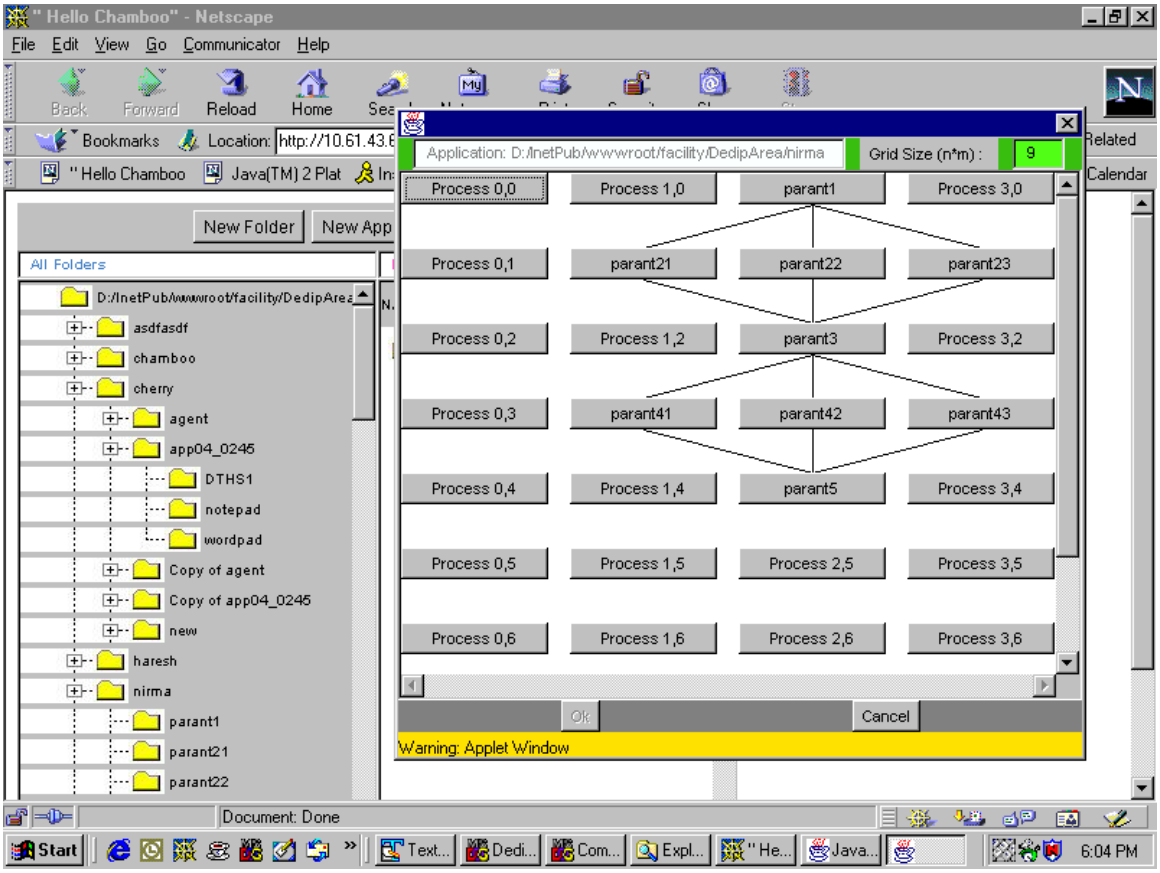


Figure 2: DEDIP GUI to configure the parallel application

The task Parant21, Parant22 and Parant23 are the copies of Parant2 running in parallel. The same is true for parant4 also. The DEDIP GUI was used to provide the information about remote node on which the process is to be executed.

Following type of modifications were carried out at a few places for satisfying the DEDIP requirements. It was possible to complete them in one day.

Original code

```
fp = fopen("A1.dat", "r");
```

Modified code

```
{
char DedipArea[150], ApplicationName[50];
```

```
 int counter;

strcpy(DedipArea,argv[argc-2]);

strcpy(ApplicationName,argv[argc-3]);

counter = atoi(argv[argc-1]);

fp = fileopen ("A1.dat", "r", DedipArea, ApplicationName, counter, "Intermediate");

}
```

The DedipArea is the logical working area in which the temporary files will be stored on any remote machine. The physical area will differ from one system to another. ApplicationName is required to find out the actual path of the application to which process belongs. Counter is used to provide the SAMD facility to the application. The intermediate and output files created will become unique by using the method incase the same application is started multiple times for different dataset. The argument decides whether the file is input file (Constant), intermediate file or the output file. DEDIP will delete all the intermediate files on successful completion of the application. DEDIP delete output files periodically at define by the DEDIP administrator.

The application was executed using the DEDIP run time environment.

4.2 Web applications

Another usage of the DEDIP was in the field of developing web based applications. Recently, DEDIP was used by scientists were for developing web-based applications for automating several tasks using Intranet [14].

4.2.1. Different applications

The scientists needed to develop various web-based applications like hierarchical progress reporting & compilation, meeting management, project task management, personal task management, document authentication, resource booking, complaint management, job workflow, remote system configuration detection, etc.

These applications have front-end browser based GUI to interact with the user. Each application has the business logic at the back-end. Each application needed to store & retrieve the information into the database on the database server. The application needed to create the required web pages dynamically on the web server and link them to the application. Some applications required connecting to the email server for sending information to the user in electronic form. Thus each application follows web based client-server model where the back-end business logic is executed on different servers while the front-end GUI is executed on the user node through the standard browser.

4.2.2. Solutions

Each application had complex GUI that cannot be implemented using simple HTML. Active Server Pages were also not found adequate for the application. Hence it was decided to opt for Java applets. The server side execution as well as inter server communication can be supported by CGI scripting or Java servlet. It needs to work out communication protocol, server to server interface and tedious coding for data communication for each application. Furthermore, it would restrict the modifiability due to complexity involved in development. The protocol and network communication may be required to be changed every time a new functionality is added in an application.

Initially, a feasibility study was carried out for DEDIP usage to reduce the application development complexity. On success, all the applications were developed using the DEDIP.

4.2.3. DEDIP Usage

The DEDIP has three major components; DEDIP server, DEDIP agent on every machine and browser based GUI. The DEDIP GUI was not used as each application has its own complicated browser based GUI. The DEDIP server and agents were found useful for executing business logic on different servers. The generalized implementation model is shown in figure-3.

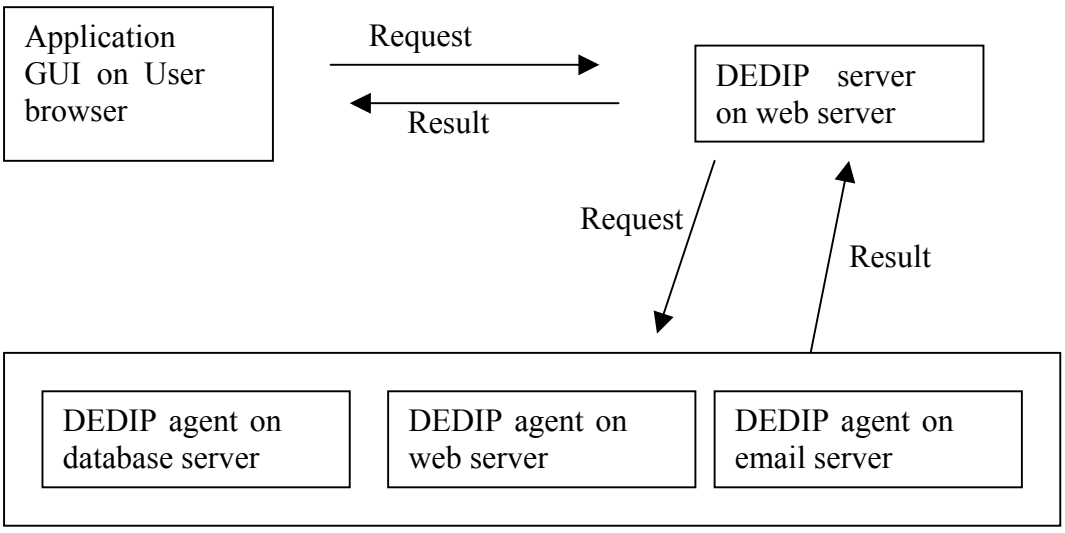


Figure 3: Application implementation model interfacing DEDIP components

The application GUI communicates directly with DEDIP server. It requests the DEDIP server to execute its business logic on the specific node. The DEDIP server

passes the request to the DEDIP agent on the specific node. The DEDIP agent satisfies the requests and returns the result to DEDIP server. DEDIP server returns the result to the application GUI.

The DEDIP has its own library for communication among DEDIP server, DEDIP agents and DEDIP GUI. The application designers were asked to use a class named "RequestObject". They need to call only one function "RequestToServer(remotenode, Object)" to interface with server. The passed-object implements the business logic. The user should develop this class based on his application requirement. He needs to implement an interface called "void ExecuteOn Server()" where he controls the business logic. He need not bother about the client-server programming, communication protocol, and server-to-server communication. He simply has to interface his business object with his GUI.

When client GUI invokes RequestToserver(….) function, it passes application object to DEDIP server residing at the web server. The DEDIP server passes application object to the required agent at "remotenode". The agent executes the function "ExecuteOnServer()" of the application object. It returns the application object back to DEDIP server. The DEDIP server returns the application object back to the application GUI. The returned object contains status as well as the data generated by the user's business logic.

All this communication and execution is transparent to the user. It is carried out by DEDIP.

The user has to simply develop an interface with DEDIP. Each application designer could easily understand the interface and adept it within an hour.

DEDIP needed a small modification in the communication library for such an interface. This modification was carried out within one day.

5. Conclusion

It is observed during the implementation of the structural analysis algorithm for civil engineering using DEDIP that the user had to modify his code as required by the DEDIP. However, the modification is quite simple, easy to understand and easy to implement. On the other hand, the parallel programming using an environment like PVM would have been too complex. As the analysis model taken for the test case was quite simple, we could not compare the computation timings for sequential processing and parallel processing. However the ease of implementation has encouraged as well as given confidence in using DEDIP for complex analysis algorithms.

The web based application had really proved the DEDIP capabilities in simplifying a complex client-server implementation. DEDIP is currently being used by 10 to 15 web based applications. The users have started using DEDIP for other applications, which are under development.

Acknowledgement

Authors thank Mr. Pranav Vora for his implementation.

References

1. http://www.epm.ornl.gov/pvm
2. M. Philippsen, M. Zenger, JavaParty – transparent remote objects in Java, Proc. of ACM 1997 PPoPP Workshop on Java for Science and Engineering Computation, (1997).
3. T. Brecht, H. Sandhu, M. Shan, J. Talbot, ParaWeb: towards world-wide supercomputing, Proc. of 7th ACM SIGOPS European Workshop, (1996).
4. Baratloo, M. Karaul, Z. Kedem, P. Wijckoff, Charlotte: Metacomputing on the Web, Future Generation Computer Systems, Vol. 15, (1999), 559-570.
5. N. Camiel, S. London, N. Nisan, O. Regev, The POPCORN project: Distributed Computation over the Internet in Java, 6th International World Wide Web Conference, (1997).
6. M. Neary, B. Christiansen, P. Cappello, K. Schauser, Javelin: Parallel computing on the internet, Future Generation Computer Systems, Vol. 15, (1999), 659-674.
7. L. Sarmenta, Bayanihan: Web-Based Volunteer Computing Using Java, 2nd International Conference on World Wide Computing and its Applications, (1998).
8. H.Takagi, S. Matsuoka, H. Nakada, S. Sekiguchi, M. Satoh, U. Nagashima, Ninflet: a Migratable Parallel Objects Framework using Java, Proc. of the ACM 1998, Workshop on Java for High_performance Network Computing, Palo Alto, CA, (1998).
9. T. Haupt, E. Akarsu, G. Fox, W. Furumanski, Web based metacomputing, Furture Generation Computer Systems, Vol. 15, (1999), 735-743.
10. Haresh Bhatt, CVS Prakash, A K Aggarwal, DEDIP: Development Environment for Distribute Image Processing, Submitted to DS Online, http://computer.org/channels/ds/
11. Haresh Bhatt, V H Patel, A K Aggarwal, Web enabled client-server model for development environment of distributed image processing, accepted in GRID-2000, International conference on metacomputing to be held at Banglore, India, during 17-20 December, 2000.
12. Ghali, A. and Neveille, A. M. : "Structural Analysis – Unified, Classical and Matrix Approach", Chapman and Hall, 1989.
13. Umesha, P. K. and Venuraj, M. T.: "structural Design Optimization with Parallel Sensitivity Analysis on Message Passing Systems", Advances in Structural Engineering - Proceedings of the International Conference on Structural Engineering, Ghaziabad, pp. 704-710, Sept. 1999.
14. Design overview of project & work-flow management automation, Technical report, CNF/SIIPA, Space Applications Centre, Ahmedabad, India.