

# Web enabled client-server model for development environment of distributed image processing

Haresh S. Bhatt \*, V. H. Patel\* and A. K. Aggarwal §

\* Space Applications Centre  
Jodhpur Tekara,  
Satellite Road,  
Ahmedabad – 380 053  
INDIA  
{haresh,vhpatel} @ipdpg.gov.in

§ Department of Computer Science  
Rollwala Computer Centre  
Gujarat University  
Ahmedabad – 380 009  
INDIA  
aka19@hotmail.com

**Abstract.** Image processing applications (IPA) requirements can be best met by using the distributed environment. The authors had developed an environment over a network of VAX/VMS and Unix for distributed image processing. The efficiency was as high as 90-95%. This paper presents an augmentation and generalization of the environment using Java and web technology to make it truly system independent. Although the environment has been tested using image processing applications, the design and architecture is truly general so that it can be used for other applications, which require distributed processing.

**Keywords:** DEDIP, Parallel Image Processing, Distributed image processing

## 1. Introduction

Image processing applications (IPA) require processing on large volumes of data. These also require various types of resources like high-resolution graphic displays, drives for magnetic tape/cartridge/floppies, optical disk, database, etc. The resource requirement changes with time due to the availability of new and better resources. Hence, it is not possible to assume the availability of all resources on a single system. The distributed processing environment not only helps in optimum utilization of such resources but also helps in achieving better throughput using multiple processors in parallel. However, if one has to use multiple heterogeneous machines in a network to execute a set of tasks, one may have to execute a tedious set of commands. It is not possible to expect an operator to carry out such operations on a regular basis. Moreover in such a system, any error, occurring either during data transfer or during processing, creates difficulties for the operator.

Development of an application having built-in automated data transfer, capability of using multiple machines in parallel, and robust error handling is a challenging job. This paper presents the authors' contribution in providing a tool that makes such a development very easy.

The research work carried out by eminent computer professionals [1-10] focused on the parallel-processing experts' needs. The image processing applications are

developed by scientists (mathematicians, physicists, remote sensing experts, etc) not by parallel processing experts. *Smooth operational environment*, *Operational setup*, and *ease of use* are the critical issues for scientists compared to parallel processing experts.

We focused on the requirement of this vast community. We presented a full fledged Development Environment for Distributed Image Processing (DEDIP) that makes the development & operationalization of distributed applications very easy [11]. This paper presents a WebDedip, which is redesign and generalization of DEDIP to make it more user friendly and truly heterogeneous.

## **2. Generalization and Extension**

The WebDedip has a novel design, which explores object oriented modeling technique in the web domain. The new model uses three-tier architecture instead of master-slave one. The DEDIP had provided GUI only on the host system. The WebDedip provides browser based GUI on all nodes connected to the system. It enables the user to use the WebDedip from any system on Internet. Thus, it provides the roaming profile to application designers, operation managers and operators. Users have to edit a few text files to configure their application in DEDIP. WebDedip has made this task easier by adding new user friendly GUIs. The augmented design addresses all the important redundancy issues making the WebDedip fault tolerant.

## **3. WebDedip overview**

The WebDedip has a three tier architecture; GUI, DedipServer and Agents, as shown in figure-1.

The GUI is the web enabled graphical user interface to make the entire user-interaction truly system independent. It supports various Java Applets for application configuration, application building, application operation initiation, application progress monitoring, and session controlling. The user initiates the interaction by visiting a predefined site using a standard browser. The standard web server loads the required GUI on the web browser.

It has a back-end DedipServer running on the web site. When the GUI submits the request to the DedipServer, it reads the application configuration information from the configuration file. The DedipServer initiates the execution of the first process in the interdependency chart. Normally, most of the applications have a single starting process. If any application has multiple starting processes, it initiates execution of all such independent processes. It informs the agent(s) on the target node to start the execution of the process. The agent sends the status information back to the DedipServer when the process is completed. The DedipServer finds out the dependent processes on the successful completion of a process and initiates the execution of each such process. The required files are transferred from one node to another. WebDedip has a callable library in Java to interface with the FTP server [12] that helps in transferring files. The required process is automatically inserted in the configuration

when IP designer inserts the IO dependency information (figure-5) between two processes.

The DedipServer stores complete information about all the applications configured on the web site. The DedipServer exchanges information with the DedipBackupServer making the model fault tolerant.

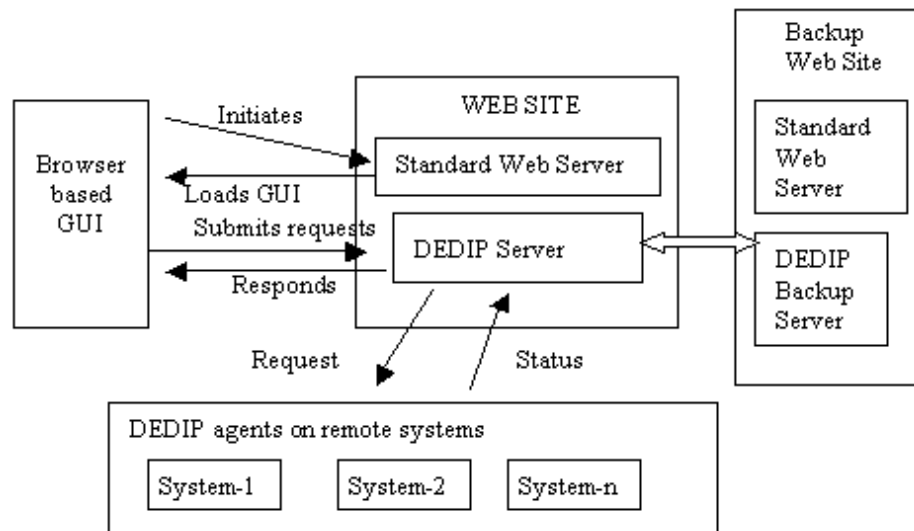


Fig. 1. WebDedip Model

The task of the agent is very simple. It accepts requests from the DedipServer, executes them and provides the status information when completed. It has process building (compilation), execution, and monitoring capabilities. It can schedule multiple processes in parallel. It does not control the synchronization among the parallel processes, instead it depends on the DedipServer for this job. It treats each process as a single independent entity.

The WebDedip not only caters to the requirements of the application designer, but also addresses all the requirements of the operation manager as well as operators. The application configuration and building is a privileged task, carried out either by the application designer or operation manager. During the regular operations, the operator can initiate any required application, monitor progress, do error handling, and terminate the application, if necessary. The web server capability is used to provide the required access control rights.

Object-oriented modeling (implemented in Java) is used for the design of the WebDedip [13]. The application is modeled as an object while the process is modeled as an embedded object. The object inter linking capability is used to maintain interdependency information for an application. Java distributed object architecture is

used along with the object serialization for network communication among GUI, DedipServer and agents. Hence, WebDedip can be used on a LAN, WAN or on Internet. Agents may run on any system over Internet. On start, an agent makes connection with DedipServer on a predefined port and volunteers for computation workload. Java object persistence is used in storing the information, including dynamic information. The same is explored in communication among the GUI, DedipServer and agents.

The Windows-explorer is used as a metaphor in developing the navigation GUI due to its popularity and ease of use (see figure-2).

### 3.1 Application Configuration

The application designer first decides the configuration of his application. It depends on the distributed resource requirement, parallel processing requirement, input/output of each process, etc. The WebDedip supports a nice GUI for the same as shown in figures 2-5. Figure-2 shows the overview of the application. The typical interdependency chart, generated interactively, is shown in figure-3. The detailed information about each process is shown in figure-4 for a typical process. The line joining two processes shows their interdependency in top-down model. The IO dependency, if any, is a part of this interdependency and it can be easily configured. The typical configuration is shown in figure-5. User can modify his application configuration file any time. The effect of the modification will be applied on next execution of the application.

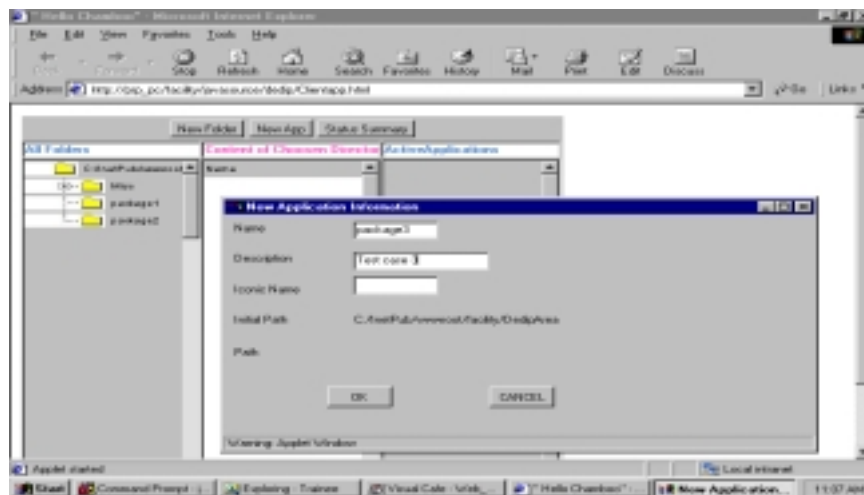
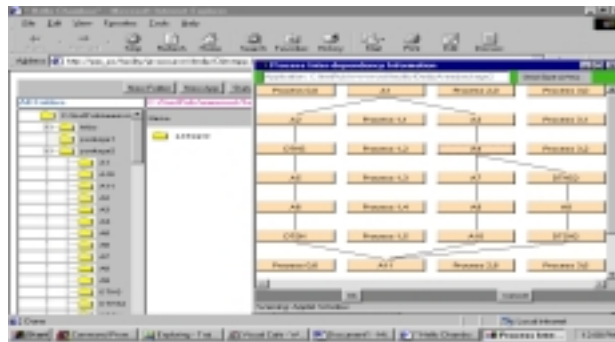


Fig. 2 Basic Information of the Application



**Fig. 3** Process interdependency Information

The screenshot shows the 'Process Detail Window' dialog box. It contains several input fields for configuring a process. The fields include:
 

- Process Name: [Text field]
- Process ID: [Text field]
- Process Type: [Dropdown menu]
- Resources: [Text field]
- Terminal ID: [Text field]
- File: [Text field]
- Depends on: [Text field]
- Dependent Processes: [Text field]
- Resolution Type: [Text field]

 At the bottom, there are 'OK', 'Cancel', and 'Apply' buttons.

**Fig. 4** Process Information

The screenshot shows the 'Data Dependency Information' dialog box. It includes fields for 'Source Process Name', 'Destination Process Name', and 'Number of Files'. Below these is a table with columns for 'Source File Name', 'Destination File Name', and 'Type'. The table lists four entries, all with the same source and destination file names and a 'Binary' type.

| Source File Name                                     | Destination File Name                                | Type   |
|--|--|--------|
| C:\Program Files\Microsoft Office\Office12\MSO12.BIN | C:\Program Files\Microsoft Office\Office12\MSO12.BIN | Binary |
| C:\Program Files\Microsoft Office\Office12\MSO12.BIN | C:\Program Files\Microsoft Office\Office12\MSO12.BIN | Binary |
| C:\Program Files\Microsoft Office\Office12\MSO12.BIN | C:\Program Files\Microsoft Office\Office12\MSO12.BIN | Binary |
| C:\Program Files\Microsoft Office\Office12\MSO12.BIN | C:\Program Files\Microsoft Office\Office12\MSO12.BIN | Binary |

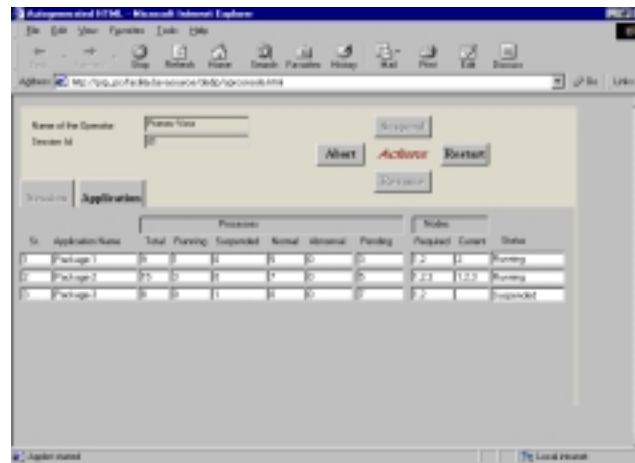
**Fig. 5** Data dependency information

### 3.2 Application building

An application consists of many processes. All these processes need to be compiled on the target node. The WebDedip has automated all these compilation. The configuration information has all the required details about each process. The DedipServer copies the source code & make-file, required to build a process, on the target node in a predefined temporary area. It then requests the agent on the node to build the process using the make-file. It carries out this task for each process given in the configuration. The agent creates designated directory and preserves the executable in it. The application designer can build the processes externally on all systems in case he is not willing to give the code. The GUI provides necessary support for such external readiness indicator.

### 3.3 Application execution and monitoring

The operator can start execution of any application from any machine on the net using the standard browser. GUI displays the configured applications to the operator for selection. Operator can start/abort/suspend/resume an application. Figures 6 & 7 show the GUI for session and application progress information.



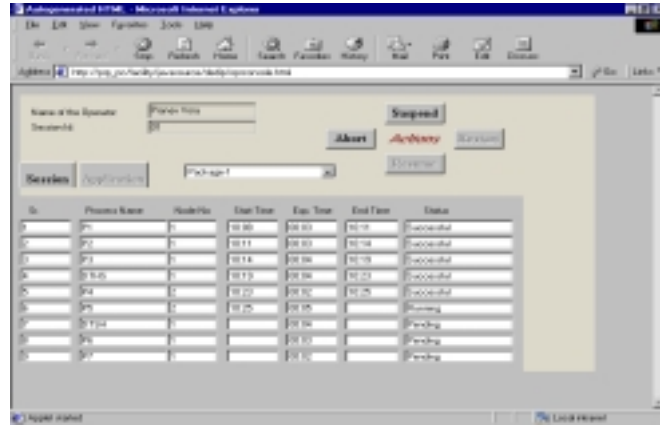
The screenshot shows a web browser window with a menu bar (File, Edit, View, Options, Tools, Help) and a toolbar. The address bar shows a URL. The main content area has a form with fields for 'Name of the Operator' and 'Device Id', and buttons for 'Suspend', 'Abort', 'Restart', and 'Refresh'. Below the form is a table with columns for 'Application Name', 'Total', 'Running', 'Suspended', 'Normal', 'Abnormal', 'Pending', 'Package', 'Count', and 'Status'.

| Sl. | Application Name | Processes |         |           |        |          |         | Nodes   |       | Status    |
|-----|------------------|-----------|---------|-----------|--------|----------|---------|---------|-------|-----------|
|     |                  | Total     | Running | Suspended | Normal | Abnormal | Pending | Package | Count |           |
| 1   | Package 1        | 0         | 0       | 0         | 0      | 0        | 0       | 2       | 1     | Running   |
| 2   | Package 2        | 0         | 0       | 0         | 0      | 0        | 0       | 2       | 1     | Running   |
| 3   | Package 3        | 0         | 0       | 1         | 0      | 0        | 0       | 2       | 1     | Suspended |

Fig 6 Session progress information

### 3.4 Error Handling

In case of abnormal completion, the DEDIP Server displays the error message with error code to the operator. These error codes and error messages are provided by application designer. WebDedip keeps this information in the configuration file. The operator can restart the process after taking the necessary actions. In addition, the operator has the options of either restarting the entire application or aborting it.



**Fig 7** Application progress information

### 3.5 Session management

Each time an operator logs in, DEDIP scheduler starts/restarts a session for him. Each session has a unique session identification number. It keeps all the information about the session on the server. The operator has multiple options to log out. He can close the session, terminate the session, suspend/resume the session, or submit the session for progress in background before logging out. He can close the session only after normal completion of all the requests he has submitted. He can terminate the session immediately in case of emergency. In case of termination, the WebDedip kills all the processes of all the requests submitted by the operator irrespective of the status. The background processing is very effective in the case of non-interactive processes. The WebDedip gives the detailed status to the operator at the next logon.

### 3.6 WebDedip system management

The WebDedip system consists of a DedipServer and agents. The DedipServer can detect the agent termination. It displays the message on operators' console as well as operation manager console.

The DedipServer is the most important process in the entire system. Its failure, for example, due to system crashing, can cause a severe problem. DedipBackupServer is designed to handle the failure of the DedipServer. The software package DedipBackupServer runs on the machine of the backup server and duplicates the required information from the DedipServer. An agent sends a trigger to DedipBackupServer when it fails to communicate with the DedipServer. The DedipBackupServer validates the DedipServer failure. It takes over the complete responsibility from that moment onwards and informs the operation manager. The

servers are exchanging information only in case of external events like termination of process, start of new process, initiation of an application by the operator, the start of new session, etc. The frequency of such possible events is very low. Furthermore, the volume of the information is negligible. Hence, the communication overhead for maintaining the back-up server is very low.

#### 4. Case study

WebDedip functionality and efficiency was tested using Microsoft NT as host and IRIS workstations as slaves. IIS 4 was used as web server. The front-end GUI is tested on two most popular browsers IE and Netscape

The WebDedip was tested for three cases [11] using simulated executables by three operators in ten runs. The simulated processes were generated resembling actual processes for image processing interaction/processing. The process dependency chart is given in figures 8-10. The processing node is shown in the bracket if it is different than host. DTHS stands for Data Transfer from Host to Slave while DTSH stands for the reverse process. 'T' indicates the tape unit requirement by the process. 'W' & 'W2' indicates that the process is scheduled on workstation1 and workstation2 respectively. The time (in minutes) required by each process is shown in bracket.

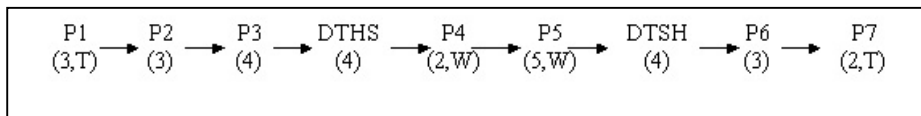


Fig. 8. Simulated case-1 for testing

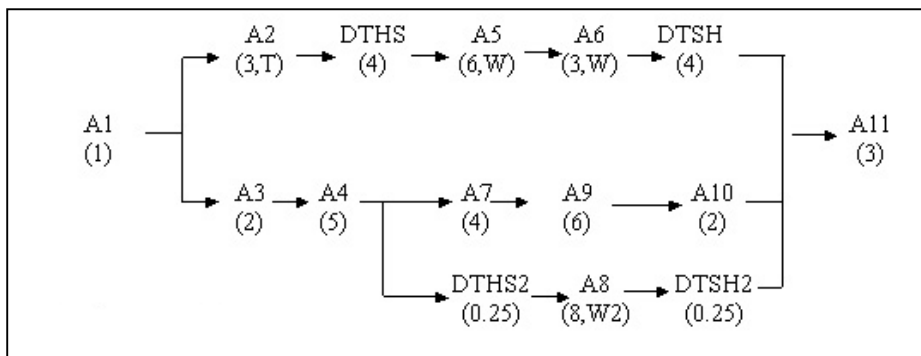
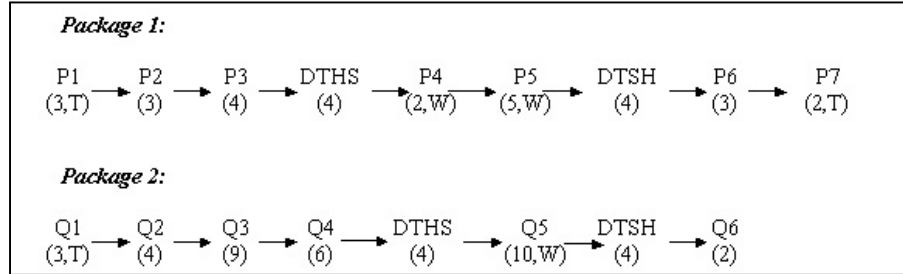


Fig. 9. Simulated case-2 for testing





**Fig. 10.** Simulated case-3 for testing

*Case 1:* Single package requiring sequential scheduling is shown in figure 8 depicting the simplest case.

*Case 2:* Single package requiring parallel scheduling is shown in figure 9.

*Case 3:* Parallel execution of two packages, each package requiring sequential scheduling, is shown in figure 10.

| <b>Table 1 : Results for the case studies (time in minutes)</b> |                    |                      |                 |
|---|--------------------|----------------------|-----------------|
| <b>Case</b>   | <b>Theoretical</b> | <b>Without Dedip</b> | <b>WebDedip</b> |
| 1   | 30.0               | 32                   | 32.0            |
| 2   | 23.5               | 52                   | 25.0            |
| 3   | 42.0               | 74                   | 46.0            |

The efficiency results are almost the same as those achieved in the earlier version, ie 90-95%. The page & applet loading time over the network is excluded. The access time in case of WebDedip is mainly due to two reasons: (1) action communication delay and (2) DedipServer overheads. This action communication delay was measured for various actions by repeated exercises. It was found out to be approximately 10 to 40 seconds on this type of action. The remaining are the DedipServer overheads.

Recently, a few scientists were engaged in developing web-based project management & work flow applications [14] like hierarchical progress reporting & compilation, meeting management, project task management, personal task management, document authentication, resource booking, complaint management, job work flow and remote system configuration detection. These applications were having distributed processing requirement amongst the browser based remote machines, web server, database server and mail server. They used WebDedip instead of Java servlets. They used their own GUI could communicate to DedipServer and agents using RequestObject, a message passing object from WebDedip library [13].

## 5. Related Work

In this section, we summarize the research efforts that are closely related to our work. JPVM [1], and Java MPI [2] are the Java extensions of PVM and MPI respectively. JavaParty [3], ParaWeb [4], Charlotte [5], Popcorn [6], and Javelin [7] are Java based systems for distributed computing using Java. JavaParty provides mechanisms for transparently distributing remote objects. ParaWeb is an implementation of the JVM that allows Java threads to be transparently executed remotely. Charlotte provides 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 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 market that determines which CPU seller executes the computelet. Javelin is an infrastructure for Internet based parallel computing. Any free computer system can volunteer to execute a task using the applets supported by the Javelin. It follows a client-broker-server architecture. Bayanihan [8] and Ninplet [9] 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 [10] is closest to our work. It provides Java-Swing based visual programming environment for metacomputing using Java. It supports Globus [15] metacomputing toolkit at the backend.

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

WebDedip (& DEDIP) concentrated on the vast community of scientists rather than efficient programmers. 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.

## 6. Conclusion

The WebDedip provides a useful facility to the designer to develop the distributed image processing application in a user-friendly environment. The browser based GUI enables him to use the system functionality from anywhere over the Internet. The graphical user interface makes it easy to visualize and configure the application.

Furthermore, WebDedip addresses all the critical elements for smooth operations. The option of back-up server support makes the entire system robust.

The results obtained from the simulated test cases for the WebDedip match with those of the earlier version. The communication delay over the network is the only additional delay. The earlier version of the model was used by 15 scientists for development and operationalization of 10 distributed image processing applications for Indian Remote sensing Satellite (IRS). The same is likely to be replaced by the new WebDedip.

Although the WebDedip has been tailored for the requirement of image processing applications, its design and architecture is truly general so that it may be used for other applications also. Collaboration is being worked out with Nirma Institute of Technology to use WebDedip in field of advanced computing for civil engineering.

A study is being carried out for interfacing WebDedip and PVM.

## References

1. A.J. Ferrari, JPVM - *The Java Virtual Machine*, <http://www.cs.virginia.edu/ajf2j/jpvm.html>.
2. S. Taylor, *Prototype of Java-MPI package*, Available at <http://cizr.anu.edu.au/sam/java/jav.mpi.prototype.html>.
3. M. Philippsen, M. Zenger, *JavaParty – transparent remote objects in Java*, Proc. of ACM 1997 PPOPP Workshop on Java for Science and Engineering Computation, (1997).
4. T. Brecht, H. Sandhu, M. Shan, J. Talbot, *ParaWeb: towards world-wide supercomputing*, Proc. of 7<sup>th</sup> ACM SIGOPS European Workshop, (1996).
5. Baratloo, M. Karaul, Z. Kedem, P. Wijckoff, Charlotte: *Metacomputing on the Web*, Future Generation Computer Systems, Vol. 15, (1999), 559-570.
6. N. Camiel, S. London, N. Nisan, O. Regev, *The POPCORN project: Distributed Computation over the Internet in Java*, 6<sup>th</sup> International World Wide Web Conference, (1997).
7. M. Neary, B. Christiansen, P. Cappello, K. Schauser, *Javelin: Parallel computing on the internet*, Future Generation Computer Systems, Vol. 15, (1999), 659-674.
8. L. Sarmenta, *Bayanihan: Web-Based Volunteer Computing Using Java*, 2<sup>nd</sup> International Conference on World Wide Computing and its Applications, (1998).
9. H. Takagi, S. Matsuoka, H. Nakada, S. Sekiguchi, M. Satoh, U. Nagashima, *Ninplet: a Migratable Parallel Objects Framework using Java*, Proc. of the ACM 1998 Workshop on Java for High-performance Network Computing, Palo Alto, CA, (1998).
10. T. Haupt, E. Akarsu, G. Fox, W. Furumanski, *Web based metacomputing*, Future Generation Computer Systems, Vol. 15, (1999), 735-743.
11. Haresh Bhatt, CVS Prakash, A K Aggarwal, *DEDIP: Development Environment for Distributed Image Processing*, Submitted to DS Online, <http://computer.org/channels/ds/>
12. *Java based client object library for interfacing with FTP servers*, Technical report, DWPIP project, (1999).
13. *Design of Web based DEDIP using UML*, Technical report, DWPIP project, (2000).
14. *Design overview of project & work-flow management automation*, Technical report, CNF/SIIPA, Space Applications Centre, Ahmedabad, India.
15. *Globus project*- <http://www.globus.org/>