Chapter 8

Web enabled client-server DEDIP

1. Introduction ................................................................. 2
2. Model ........................................................................ 3
  2.1 Navigation ................................................................ 5
  2.2 Application Configuration ....................................... 5
  2.3 Application building .................................................. 11
  2.4 Application execution and monitoring ......................... 11
  2.5 Error Handling ........................................................... 13
  2.6 Session management ................................................... 14
  2.7 DEDIP system management ........................................ 14
3. Utilities .......................................................................... 15
  3.1 File transfer ............................................................. 15
  3.2 Application Results storage ........................................ 16
4. Overview of analysis and design ..................................... 16
5. Case study ..................................................................... 16
6. DEDIP usage for web application development ............... 18
7. Conclusion .................................................................... 20
1. Introduction

This research project aimed at studying the image processing software optimization for fulfilling its processing requirements. The objective of this research work was not confined to the concept proving implementation of new ideas but was aimed at providing necessary tools. Hence, A full-fledged Development Environment for Distributed Image Processing (DEDIP) was not only conceptualized but also operationalized [VI]. It is discussed in detail in chapter 7.

"Ease of use" was the main theme behind this application oriented research work. Hence, the efforts were not required up to software operationalization. The survey concluded that the tool has proved the usefulness for IRS-1C. However, the tool needs to fulfill following requirements to make it more useful and easier:

- The model supports the VAX/VMS and Unix operating systems. It should be made truly system independent.
- The users are required to carry out few tedious and error prone tasks. The aim of this research work was to provide a tool that is easy to use. Hence, it needs to be augmented to make the model more usable.

Users need to edit application configuration file for giving the process interdependency information. They have to strictly follow the predefined format. They need to study the format and adhere to it. Furthermore, the DEDIP supports transfer of the required file from one node to another as per application need. The user needs to provide the list of such files in a predefined format. User needs to insert the DTHS or DTSH programs into his application configuration at proper place as seen in figure 7.1. It can be easily visualized in graphical mode but user found it difficult in editing the same in predefined format. It would be better to make this interface easier through the visual interface. We adopted the GUI usage for parallel compilers from [38] and extended for configuring parallel and distributed applications under DEDIP.

- Users need to build their executables on all the target nodes and keep them at predefined locations. Hence, users have to carry out this tedious work whenever they modify their applications. The DEDIP also decided to support the automatic application building incase the user provides the required source code and make-
Web enabled DEDIP

file. PVM [39] also supports such facility. We support parallel compilation in contrast to PVM.

- The DEDIP depends entirely on the host system in its master-slave architecture. Hence, the host system failure is a bottleneck. The DEDIP provided the facility to continue with the previous session with the minimum loss of processing power. However, it needs the host system up continuously. It has to wait for failure diagnosis and recovery action.

Internet technology has proved its usage not only in information area but also in workflow automation too. This research project decided to study whether the Internet technology is useful for distributed image processing or not. It was found that DEDIP can provide more flexibility for user interaction using the Internet technology.

A study was also conducted for usage of Java in making the DEDIP system independent.

It was decided to make the model more generic covering wider scope of usage.

A new architecture for DEDIP was worked out exploring object oriented modeling technique in the web domain. It is a three-tier architecture instead of master-slave model. The browser based GUI provides the roaming profile to application designer, operation manager and operators. It addresses the above requirements for providing ease of use. The augmented design addresses all the important redundancy issues making the model fault tolerant. Although the main aim was to provide an environment for image processing applications, the design and architecture is truly generic for use by other applications of a similar kind.

This chapter discusses the new architecture of the DEDIP. Section 2 briefly describes the model along with new facilities.

2. Model

The web based DEDIP model is a three tier architecture; DEDIP GUI, DEDIP Server and DEDIP agents, as shown in figure-8.1. The task of GUI, server and agent is same as OPRINT, HostManager and SlaveManager in chapter-7. The DEDIP server is having additional responsibility of catering to supply data as required by GUI. The task of the GUI is restricted to the user-interaction.
The DEDIP GUI is the web enabled graphical user interface making the entire user-interaction truly system independent. It supports various forms 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.

Figure-8.1: Client-Server model of Web Dedip

It has at the back-end the DEDIP server running on the web site. The DEDIP GUI submits the task to the DEDIP server. The DEDIP server initiates the execution of the tasks as per the configuration information. It requests the remote agents to schedule the process. It monitors the entire session’s progress. The DEDIP server maintains complete information about all the applications configured on the web site. The DEDIP server exchanges information with the DEDIP backup server making the model fault tolerant.
The task of the DEDIP scheduler-agent is very simple (same as SlaveManager). It accepts requests from the DEDIP server, 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 DEDIP server for this job. It treats each process as a single independent entity.

The DEDIP not only caters to the requirements of application designer, but also addresses all the requirements of the operations manager as well as operators. The application configuration and building is a privileged task, carried out either by the application designer or operations manager. During the regular operations, the operator can initiate any required application, monitor progress, cope with error handling, and terminate the application, if necessary.

Object-oriented modeling (implemented in Java) is used for the design of the augmented DEDIP [DWPIP-3]. 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. The object serialization is used in storing the information, including dynamic information. The same is explored in communication among the DEDIP GUI, server and agents.

2.1 Navigation

A simple blank Java applet is loaded on the browser by the web server. The applet intern makes connection with DEDIP server. It gets the list of applications that are configured in DEDIP. It displays the list in a tree view. The Windows-explorer is used as a metaphor in developing the navigation GUI due to its popularity and ease of use (see figure-8.2). The user can configure a new application, build a configured application or start a ready application using the navigation GUI.

2.2 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 DEDIP supports a nice GUI for the same as shown in figure-8.2. He first provides the overview

8.5
Web enabled DEDIP

information about the application using the GUI shown in Figure-8.2A. Then he configures his process interdependency chart using GUI shown in figure-8.2B. The line joining two processes shows their interdependency in top-down mode. This GUI is quite user friendly giving the complete visualization of the process interdependency. It is as simple as drawing the chart on a paper. Furthermore, user can easily visualize the insertion point of DTSH and DTHS as per the file transfer requirement.

Each button in the figure 8.2B represents a process of the application. The user gives the detailed information about the process as shown in figure-8.2C.

The list of files to be transferred can be given using the GUI in figure-8.2D.

When the GUI submits application configuration to the DEDIP server it stores all the information on the web site in the predefined format. The user can easily modify the configuration, as and when required, using the same GUI. The application designer can provide the rights to the operations manager to modify the application configuration, if required. A directory structure is designed for storing all the information. The DEDIP creates and modifies the structure during the application configuration. The directory structure can be visualized from figure 8.2A and 8.2B.
Web enabled DEDIP

Figure 8.2A: Basic Application Information Form
Web enabled DEDIP

Figure 8.2B: Application Configuration Information Form
Web enabled DEDIP

Figure 8.2C: Process Information Form
Figure 8.2D: Data dependency Information Form
2.3 Application building

An application consists of many processes. All these processes are required to be compiled on the target node. The DEDIP automates all such compilation. The configuration information has all the required details about each process. The DEDIP directory structure is designed (and automatically created during the application configuration) to store the source-code & make-file(s) on the server.

The DEDIP server copies the source code & make-file, required to build a process, on the target node in a predefined temporary area. It then requests DEDIP 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 DEDIP agent preserves the executable in the designated directory. The DEDIP agent has the capability to create the required directory structure. The detailed build status is stored in configuration file on the server. The DEDIP server allows the application execution only when all the processes are built successfully on the target nodes.

The application designer can build the processes externally on all systems incase he is not willing to give the code. The GUI provides necessary support for such external readiness indicator.

2.4 Application execution and monitoring

The operator can start execution of any application from any machine on the net using the standard browser. DEDIP GUI displays the configured applications to the operator for selection. When the operator submits the request to the DEDIP server, it reads the application configuration information from the configuration file. The DEDIP server 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 DEDIP agent(s) on the target node to start the execution of the process. A DEDIP agent can be installed on the server also, to use the server as a processing node. The DEDIP agent sends the status information back to the DEDIP server when the process is completed. The DEDIP server finds out the
Web enabled DEDIP

dependent processes on the successful completion of a process and initiates the execution of each such process.

Figure-8.3 shows the GUI for session and application progress information.

Figure 8.3A: Session Progress Information
2.5 Error Handling

In case of abnormal completion, the DEDIP Server displays the error message with error code to the operator. Each application designer has to provide the error codes along with the corresponding meaningful message text. DEDIP maintains this information in the configuration file.

The operator can restart the process after taking the necessary actions. In addition, the operator has the options to restart the application or to abort the application.
DEDIP maintains complete information about the process termination. This enables the operator to carry out the error handling during the next logon also. Often the problem in this mode is the disk space occupied by the intermediate files. DEDIP sends the message to operation manager in such a case.

2.6 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 maintains 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 submitted by him. He can terminate the session immediately in case of emergency. The DEDIP 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 applications. The DEDIP gives the detailed status to the operator at the next logon.

2.7 DEDIP system management

The DEDIP system consists of a DEDIP server and DEDIP agents. The server makes a connection with all the required agents at predefined time interval. Hence, it is able to detect the crashing of any computer. It displays the message on operators' console as well as sends message to operation manager.

The DEDIP server is the most important process in the entire system. Its failure, for example, due to system crashing, can cause a severe problem. The DEDIP design supports the configuration of a backup server. The operation manager can install the DEDIP server on any machine as backup of the main server. The DEDIP server updates the back-up server on each important event. The DEDIP server can take over the complete responsibility at any time in case the main DEDIP server fails. The DEDIP agents help in such a take over. The DEDIP agent normally makes three attempts (at specified time interval) to pass the process termination status to the DEDIP server. Incase, it
fails after three attempts, it contacts the DEDIP backup server. The backup server validates the main server failure and informs the operations manager. The operations manager can make the back-up server as the main server. At the same time, he can start another backup server too, if installed.

The operation manager can toggle the main and back-up server irrespective of any failure condition. This provides him full control over the entire system. The operation manager also has browser based GUI to perform these task.

The servers are exchanging information only incase of external events like process termination, a new process starts, operator initiates an application, new session starts, 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.

3. Utilities

The DEDIP supports and uses the following important utilities:

3.1 File transfer

The image processing application requires a large volume of data transfer across the distributed processors. The DEDIP system supports automatic assured data transfer mechanism. It makes three attempts failing which it asks for operator help. A general-purpose data transfer process is developed for the same. The process is automatically inserted in the configuration when IP designer inserts the I/O dependency information (figure-8.2D) between two processes.

The DEDIP provides the callable library in Java, which can be useful to the application designers for self-controlled data transfer. The library makes interface with the standard FTP servers for actual data transfer [DWPIP-3].
3.2 Application Results storage

The DEDIP directory structure contains the predefined location for the output of each application. The operation manager can configure the DEDIP root path based on the disk space availability. The users are requested to create their output in the predefined location. The user can access his results from any system on net using the standard browser. The web server provides the access rights to the user only for his application area.

4. Overview of analysis and design

The most advanced version of object oriented technology, the UML, is used for analysis and design of the final model. The detailed analysis and design is given in [DWPIP-3]. Sun had worked out delegate-model concept from Netscape’s model-view-control concept. It helps to isolate the basic model from the user interaction. The new DEDIP followed the same. The DEDIP has various model classes as well as GUI classes.

Appendix-4 contains major UML diagrams along with the brief description about major classes.

The object serialization gave flexibility to the DEDIP designers to generate the dynamic protocol for interaction between various GUI, DEDIP server, backup server and agents. The object persistence helps in storing the important information; for example application configuration.

5. Case study

The main aim of the DEDIP was to provide a tool that is easy to use for developing a distributed parallel image processing application. The enriched GUI is supported at all places for application configuration, execution and monitoring. It can be best visualized from the screen shots given in figure 8.2 and 8.3. The Windows NT and IIS4 were used as web server for testing on the 10 Mbps Intranet. The front-end GUI is tested on two most popular browsers IE and Netscape supporting Java-2 plugins.

DEDIP functionality and efficiency was tested using Microsoft NT as host and IRIS workstations as slaves.
Web enabled DEDIP

The DEDIP was tested for three cases 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 figure 8.4. The elapsed time requirement and processing node is shown in the bracket. The process DTHS indicates the Data Transfer required from host to slave where as DTSH indicates the reverse. 'T' indicates the tape unit requirement by the process. 'W' & 'W2' indicate that the process is scheduled on workstation1 and workstation2 respectively. The time (in minutes) required by each process is shown in bracket.

Case 1: Single package requiring sequential scheduling is shown in figure 8.4A depicting the simplest case.

Case 2: Single package requiring parallel scheduling is shown in figure 8.4B.

Case 3: Parallel execution of two packages, each package requiring sequential scheduling, is shown in figure 8.4C.

Figure 8.4.A: Single package with sequential execution

Figure 8.4.B: Single package with parallel processing requirement
Web enabled DEDIP

Package 1:

P1 (3,7) → P2 (3) → P3 (4) → DTHS (4) → P4 (2,7) → P5 (5,7) → DTHS (4) → P6 (3) → P7 (2,7)

Package 2:

Q1 (3,7) → Q2 (4) → Q3 (5) → Q4 (6) → DTHS (4) → Q5 (10,7) → DTHS (4) → Q6 (2)

Figure 8.4.C: Two packages executed in parallel.

Table 1: Results for the case studies (time in minutes)

<table>
<thead>
<tr>
<th>Case</th>
<th>Theoretical</th>
<th>Web DEDIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30.0</td>
<td>32.0</td>
</tr>
<tr>
<td>2</td>
<td>23.5</td>
<td>25.0</td>
</tr>
<tr>
<td>3</td>
<td>42.0</td>
<td>46.0</td>
</tr>
</tbody>
</table>

The efficiency results are almost the same as those achieved in the earlier version, i.e., 90-95%. The access time in case of DEDIP is mainly due to two reasons: (1) action communication delay and (2) DEDIP server 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 is the DEDIP server overheads.

6. DEDIP usage for web application development

The DEDIP is conceptualized and operationalized for distributed image processing applications. However, its design is generic so it can be used for other class of
applications too. It was decided to study its usability in the area other than the image processing.

Recently, few scientists were engaged in developing web-based applications for SAC Intranet [68]. They had to automate various activities like hierarchical progress reporting & compilation, meeting management, project task management, personal task management, document authentication, resource booking, complaint management, job work flow, remote system configuration detection, etc. These applications needed the server side execution for database connectivity, dynamic web page creation, interfacing with mailing server. Such a web-based application is quite complex in comparison with dynamic web sites. It is a full-fledged application giving user friendly GUI on standard browser with required functionality. Web based application development needs the client-server modeling. The DEDIP server was found to be useful for making the development easier. The DEDIP server was customized to support their server side execution requirements. These web-based applications did not require the DEDIP GUI, instead they needed direct interface with DEDIP server. The DEDIP is having 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( Object)” to interface with server. Furthermore, they need to implement an interface called “Execute On Server”. The object, passed as an argument, need to implement the server side functional components. RequestObject passes the object to DEDIP server residing at web server. The DEDIP server returns the object back to RequestObject after executing the required component. The return object may contain status as well as the data generated by the functional components. This interface is very easy. Hence, the application designer could easily adopt within an hour.

Another option with them was to use Java servlets or CGI interface with Java applets. This option would need to work out communication protocol, servlet to servlet 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 require changing every time a new functionality is added in an application. The DEDIP made their development very easy making the functional components independent of network communication.
Web enabled DEDIP

The DEDIP customization required only one day efforts. This has proved the generic nature of DEDIP.

7. Conclusion

The DEDIP 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. The graphical user interface makes it easy to visualize and configure the application. Furthermore, DEDIP addresses all the critical elements of the operation. The option of back-up server support makes the entire system robust.

The result obtained from the simulated test cases for the augmented DEDIP 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 applications. The same is likely to be replaced by the new web enabled client-server based DEDIP.

Although the augmented DEDIP has been focused on the requirement of image processing applications, the design and architecture is truly general so that it may be used for other applications also.