Chapter 2

Overview of our work

1. Introduction .................................................................................................................. 2
2. Goals and issues ........................................................................................................... 2
3. Virtual Parallel Processing ........................................................................................... 5
4. Parallel image processing ............................................................................................ 6
5. Distributed Processing and Distributed Parallel Processing ...................................... 11
6. Conclusion ................................................................................................................. 19
Overview

1. Introduction

The aim of our research study is focused on optimization of the image processing for parallel and distributed processing.

Under this research project, a study of image processing software optimization aspects was taken up first for the best and optimal use of computers. Authors wanted to set guidelines that are useful for parallel processing. The guidelines should help the users to optimize their software easily to improve the efficiency. The VAX/VMS system was extensively used by the users for image preprocessing and analysis [6,7]. The transputer network was the most popular parallel machine at that time [5,8,9]. Both were taken as the test bench for the study.

The authors also wanted to make optimization user independent such that the users can convert their programs written in conventional "C" into Parallel "C" efficiently without detailed study. The problem is quite complex and hence only single image case was considered.

Distributed processing was becoming a necessary tool for optimal use of resources and computational power. The focus of the study was to provide a user-friendly environment for developing distributed processing applications and operational environment over the network of heterogeneous systems. The environment should help users in developing the distributed image processing applications. It should be easy to use. That means, the environment should enable the users to use the evolving computer technology easily. Our work was more concentrated on distributed computing and metacomputing as they became more popular than parallel machine in recent era.

Let us start by defining our goals and issues.

2. Goals and issues

Ease of use, security, heterogeneity, efficiency, load balancing, scalability, fault-tolerance, adaptability, communication overheads, access control rights, process management, remote compilation, and application monitoring are some of the critical issues for parallel and distributed processing. Researchers have focused on either one or many of the above in their work. We have studied the work carried out by the scientists in the parallel and distributed image processing. We discussed with few scientists at Space Applications Centre about the relevance of above issues to the
Overview

image processing. We identified most important issues in the field of parallel & distributed image processing.

Efficiency:

An individual image processing application may not require very high processing power for one data set (image). However, it needs the higher processing power when operationally executed on multiple data sets. Thus, the applications are Single Algorithm Multiple Data (SAMD) type of parallel processing. Different image processing applications require different resources. It is difficult for an operational center to reserve resources for each application. Hence, the applications need to share the distributed resources optimally. Some people refer to this as performance.

Heterogeneity:

VAX/VMS systems have been used for remote sensing oriented image processing since many years. Mass storage devices like HDDT (High Density Digital Tape) and the image display devices are a few of the important special purpose hardware required in image processing. Different industries developing such hardware use different computer systems. For example, HDDTR (HDDT recorder) is used with VAX/VMS. Sita Image Processing System is developed using XENIX operating system. SGI developed 24 bit high-resolution graphics terminal using IRIS operating system. Furthermore, the operational center may not have homogeneous systems. Hence, the heterogeneous environment should be optimally utilized. Some refer to this as interoperability.

Scalability:

The operational center may add or remove the computer systems. Furthermore, the personal desktops and workstations within the organization can be used by the operational center whenever free, typically during off-hours. Some refer to it in a broader sense as adaptability or inceptive.

Access rights:

The application using the distributed environment should not have burden of login rights. Providing an account for each application is a critical administrative issue. On the other hand, the open access creates several problems while executing multiple applications.
Overview

Smooth operations:

Different scientists need to analyze many different images for various purposes. Providing dedicated resources to each individual is not only costly solution but also wastage of the resources. Hence, each operational center should support these periodic requirements. The periodic operations are carried out by the low skilled operators. It is not expected from the operator to support the SAMD requirement manually using standard commands. Hence, each application has to provide built-in support for the SAMD & resource management. Furthermore, an operator may execute different applications simultaneously. Hence, he needs an integrated operational environment to monitor & control all the applications through a single interface. Furthermore, operational environment needs each application to be robust. Some refer to this as ease of use and GUI (Graphical User Interface) at small scale for an application.

Operational Setup:

All the image processing applications are to be installed at the operational center. The installation requires building the applications' components on required nodes in the heterogeneous environment. Building a distributed application typically involves the following steps: (1) the required source files are transferred on required nodes; (2) the related compilation is made on every node; (3) executables are transferred in operational area; and (4) if errors are detected or modifications made, then all the previous steps have to be repeated. When any new node is added to the operational system, the above task is carried out for each application that is capable of using the new system. Therefore, a method to make these operations automatic is essential. This automation also addresses few issues pertaining to the scalability.

Ease of Use:

The development is being carried out by the scientists and not by the computer professionals. The scientists are using Fortran and 'C'. Some may be using C++ and Java. We conducted survey with system administrators at Space Applications Center to find out how many scientists use Java, C++, C and Fortran. Out of 150 scientists, 10 know Java & C++ and 10 are in learning phase. Most of them are computer professionals (not mathematicians, physicists, etc). Rest of them are using C and Fortran. They would like to concentrate in the area of their specialization rather than developing parallel & distributed-processing skills. The learning of any computer skill is an overhead for them. It is difficult to provide the support of
Overview

Computer professionals to each scientist. It has few issues pertaining to manpower requirements, intellectual property rights, communication overheads & gaps, and brain drain. This is a major issue if one wants to satisfy the large community.

These goals and issues are applicable to other image processing centers like National Remote Sensing Agency (NRSA), Regional Remote Sensing Service Centers (RRSSC), State Remote Sensing Centers, etc. It is also applicable to the image processing centers of different universities. At university the requirement may not be as high as the operational centers. Nevertheless, the resources are also very limited.

3. Virtual Parallel Processing

We started our work studying optimal utilization of a computer system. The massive task of developing data products software for Indian Remote sensing Satellites (IRS) was going on when the study began. It consists of various applications like radiometric correction, geometric correction, swath modeling, cloud cover estimation, data quality evaluation, etc. As a preliminary study, it was decided to study the VAX/VMS's virtual parallel processing capabilities and work out the guidelines for the software optimization. The guidelines may be established by optimizing more than one application. The guidelines should be generic in nature so that these may be applied to all the applications. The optimization task for an application is a difficult job. The generation of the generic guidelines, that are useful to many applications, is a real challenging job.

To start with, the Radiometric correction software was examined [6]. The VAX/VMS operating system, its architecture was studied in detail [10,11]. As the software was developed in Fortran, the efficiency aspect of Fortran was studied in detail [12,13]. The radiometric correction software was optimized [I]. The following conclusion were made from above study and experiment:

a) The software was being developed using Fortran. Users consider Fortran as a highly optimized compiler [12,13]. Only real time applications were using the assembly language [6]. Authors studied the assembly code generated by the Fortran compiler for the radiometric correction application. The authors saw a considerable scope for improvement. Authors implemented the most time consuming code in assembly language. The application efficiency was improved by 115%, much beyond the expectation of application designers [I].

b) The VMS based VAX-11/780 and microVAX were the most popular machines for development and operational use. The machines had separate I/O controller,
Overview

offloading CPU for processing [15]. The operating system provided the system services (APIs in today’s terminology) to use them in parallel [16,17]. Authors paralleled I/O and processing in the radiometric-correction software using triple buffers (one for input, the second for processing and the third for output). It improved the throughput by 40% [I]. Authors simulated the pointer mechanism in Fortran to avoid the internal memory data transfer among the three buffers [I].

c) The hard disk controllers were intelligent [15]. Each controller had built-in capability to keep the multiple requests in queue. It first services the request whose data is near to the disk-head. Authors found out, by experiment, that the buffer length of 192K (three requests of 64K) was the optimum. Using this guideline, the efficiency of radiometric correction application was improved by 25% compared to original implementation [I].

d) The application was using single hard disk for input and output. The separation of the same gave 10-15% efficiency [I].

Thus, the processing time for the entire software was optimized from 10 minutes to 2.75 minutes for a scene giving more than 300% efficiency [I]. Geometric-correction application and photo-processing applications were optimized to verify the above guidelines. A similar improvement in efficiency was found in both the optimizations too [I].

Please refer chapter-4 for more detail.

It was suspected that the efficiency would go down near to original (10 minutes for radiometric correction) when all the three applications are executed in pipeline mode for multiple data under operational environment. All the optimized applications were executed in the operational environment to verify the efficiency. It was found that the observations were still effective[18]. We decided to study the extendibility of above guidelines for parallel machine.

4. Parallel image processing

Another common source for higher processing power is the parallel machine. The transputer based PARAM, ANURAG and ANUPAM were the parallel machines available in India [19]. The image processing scientists in India were using parallel ‘C’, parallel Fortran and the transputer native language - OCCAM.
Overview

The worked carried out by the various scientists in using the parallel machine for image processing was studied [20-27]. The following observations were made from the study:

a) Parallel machines were not utilized with its full capabilities.

b) Systematic efforts were required to study
   - the implication of selection of topology, especially comparison between the ring and tree topologies.
   - how to isolate host overheads from the transputer internal communication overheads.
   - the usage of transputer system for real time image processing applications.
   - the quantification of the efforts required in implementing an algorithm on a parallel machine in comparison with the sequential code.
   - the comparison between parallel ‘C’ and OCCAM.
   - the aspect of parallelizing computation and communications tasks.
   - the selection of optimum buffer length for data communication.

c) Due to the large volume of the image files, data-partitioning technique is more beneficial. It was found that the coarse grain parallel processing technique is highly suitable to the image processing. Hence, it was decided that our study should be focused on the coarse-grain parallelism.

d) Users needed a tool which could help them in reducing the development efforts, provide better control and help in reducing the overhead of understanding the newer dimensions of parallel and distributed processing technology.

It was decided, as a part of doctorate work, that

- An application for parallel image processing should be implemented to study the above aspects.
- A study should be conducted to evaluate the available automatic parallelizing tools.

A few applications were studied for selecting one for the above implementation study [28]. Automatic Cloud Cover Estimation Software (ACCESS) [29] was selected for the study. ACCESS required a real time environment, high I/O and processing power. The real time processing requirement was 1.3 Mbps for IRS-1B and 14 Mbps for IRS-1C. Its sequential code was available. The actual code size was approximately 3000 lines spanned in 18 modules. As a part of this study, the application was
Overview

implemented on a transputer system having four processors, using parallel ‘C’ and OCCAM [II]. Ring, Tree, mesh and cube topologies were considered for comparison. Mesh and cube topologies are extension of the tree. The theoretical analysis shows that the communication overhead was 160% more for cube and 225% more for mesh when compared to the tree. Hence, it was decided to use ring and tree topologies. From the experiments, the following conclusions were drawn:

a) The development time was almost three folds compared to sequential coding, even after having knowledge of parallel processing and C.

b) The disk I/O time (time taken to load the data from disk to master node of the transputer network) of the application was 1 minute and 27 seconds for approximately 5 MB data. Such clear separation helped in eliminating the ambiguities in conclusion made by some scientists[20-27].

e) The ACCESS took processing time of 26 seconds for ring topology and 16 seconds for tree topology in parallel C. This would satisfy the real time requirement of the application for IRS-1B satellite. For tree topology, ACCESS implementation in OCCAM took only 8 seconds giving 5 Mbps processing speed. IRS-1C required 14 Mbps for real time processing. We sub sampled the data at alternate pixels and scan-lines, which reduced processing by 1/4th, to meet the requirements. We also verified that such sub sampling does not affect the accuracy of the algorithm.

d) The OCCAM was found 200% efficient than the parallel C.

e) Almost linear scalability was observed in case of tree topology. The same was not observed in case of ring topology due to higher inter processor communication overheads.

f) The tree topology required more concentrated efforts than ring topology due to synchronization issues. However, it provided very good efficiency.

g) A buffer length of 32 KB was found the best for ACCESS.

Please refer to chapter-5 for more detail.

Thereafter a study was conducted to evaluate the available automatic parallelizing tools [30-38]. The following conclusions were made from the study:

a) An efficient parallelization seems to be a very challenging and interesting research problem for computer professionals. There were two models adopted for the same, data partitioning and algorithm partitioning.
Overview

b) Most of the computer research carried out in the field of data partitioning techniques was either concentrated on (a) providing system dependent efficient parallel language like OCCAM, (b) Providing system independent parallel languages like parallel C, parallel Fortran, and ADAPT, (c) Automatic detection of data dependency and (d) Automatic conversion of a sequential program into a parallel one.

c) The mid-grain data partitioning has been studied extensively by the computer professionals. Automatic detection of such parallelism is catching the attention of most of the computer scientists, as it is useful to many applications.

d) An image-processing scientist normally concentrates on image partitioning. This coarse graining partitioning is easy to implement compared to mid-grain. Moreover, it is efficient and it provides better throughput.

e) The automatic parallelization in the field of coarse-grained data partitioning will be very much useful to a class of applications. Remote sensing scientist would appreciate such coarse grain automatic parallelization due to its proximity to their style of parallel programming.

f) The efficiency of the parallelism can be increased if user (program designer or remote sensing scientist) has control in selecting the parallelizable data.

For this study, we developed a directive-based translator (DBT) for image processing having single partitionable input image [III]. Image processing applications deal with one or more images simultaneously. However, only one image is considered as the prime input image. This image can be partitioned. We took transputer network as the target parallel machine.

The application designer is the best judge to decide the primary and secondary input image files. The primary file contains the data that can be partitioned. The DBT model expects application designer to insert certain directives into his sequential code. Authors thoroughly inspected 18 image-processing applications’ source-codes, each having actual code size of approximately 1000 lines, in deriving the DBT directives and algorithm. The list of the 18 applications is given in appendix-2. Only four directives were derived.

- /*$I_file$*/

This directive is to be inserted to indicate the primary input file, which can be partitioned among the workers.

2.9
Overview

• /*$! fileMS$*/

This directive is very useful when primary input file is being scanned (read) more than once.

• /*$P_file$*/

The directive to indicate the files containing auxiliary parameters. These parameters will be passed to all workers.

• /*$Header$*/

The directive to discriminate the header recorded from the actual data in the primary input file. This data will be passed to all workers.

The directives are meaningful, simple to understand and easy to use. The directives are inserted as comment lines. Hence, application code remains compatible with sequential machine.

DBT then parses the entire source code with the help of the directives and disintegrates it into various components like the data input from file, auxiliary parameters, main algorithm, resultant variables and data output to file. DBT establishes the exact relationship among these components. The DBT generates the master & slave programs by integrating the above components into proper format. It also inserts the code for data communication between the master and the slaves at appropriate places. The master program distributes the input data to multiple slaves, gathers resulting data from slaves and integrates the resulting data. The slaves apply the actual algorithm. DBT converts sequential 'C' code into parallel 'C'. DBT can be easily augmented for other parallel machine by modifying message passing library interface.

Figure 6.1 (chapter 6) shows the DBT model in pictorial form.

It was observed that the coding styles vary from one programmer to another. The DBT model was more concerned about the coding style adopted for master loop and reading the data from file. The DBT handles all the styles adopted in the 18 applications. DBT design has flexibility for the incorporation of other styles.

The ACCESS was taken as a test case to prove the model. It was verified that the ACCESS is a true representative of all the 18 image-processing programs. The DBT generated output results are almost matching with that of hand coded version.
Overview

hand-coded program was taking 22 seconds whereas the DBT generated program was taking 24 seconds for one image (5 MB data). Thus, the DBT efficiency was found to be 92-95% compared to hand-coded version. We had experienced during the parallel implementation of ACCESS that the parallel programming efforts required are 300% compared to the sequential one. Hence, we can conclude that the application designer can save 200% efforts (300% if sequential code is already available) at the cost of 5-8% efficiency loss. Furthermore, the user need not learn parallel processing details, programming techniques and tedious & cumbersome debugging procedures. He should have “know-how” of parallel processing.

Please refer to chapter-6 for more detail.

Thus, the research project succeeded in developing a generalized and efficient tool for the class of “single image processing” algorithms used for remote sensing data.

5. Distributed Processing and Distributed Parallel Processing

The distributed processing technology was emerging when the study began. It was becoming realizable due to the new inventions in the field of networking. Distributed processing was becoming a necessary tool for optimal use of resources and computational power. People started using the Parallel Virtual Machine (PVM) in place of actual parallel machine [39]. The Indian Transputer User Group (ITUG) was renamed as Advanced Computing Society.

High Performance Computing (HPC) initiative in North America, India, Australia and Europe aims at developing a generalized wide-area computing infra-structure, that can be easily used by research worker in various fields. This research project has developed DEDIP (Development Environment for Distributed Image Processing) that make it possible for a near-optimal use of distributed computing resources. DEDIP have been tested exhaustively. In fact, DEDIP is now in production use by image processing scientist.

The deliverables generated by the work is described in this section. It is compared with similar efforts in the world, during the last couple of years, in chapter 3.

The detailed literature survey was conducted in distributed processing environment [39-65].
Overview

Following conclusions were made from the above study:

- Most of the work in the distributed parallel-processing environment assumes that users are expert in parallel coding. They nicely addressed the issues of parallel processing experts. They do not address the basic need of vast community of scientists, i.e. ease of use.
- As they are addressing the need of a large and complex application, they do not address the issues regarding smooth operations in real sense.
- The future generation computer-systems expect the users to use Java as programming language.
- The coarse grain parallelism model used by ISRO had proven itself by the efficient resource usage. It also provides ease of implementation.

The authors decided to concentrate the need of vast community of scientists rather than the parallel processing experts. We aimed at providing a user-friendly environment for developing distributed applications and operational environment over the network of heterogeneous systems. It should be very easy to use. It should be useful for a long time. That means, the environment should be adaptable to the evolving computer technology. Normally, the image processing scientists are using the operator for periodic execution of their application. Hence, the environment should also address the operational issues. The application designer should be able to easily operationalize his distributed application. The operator should be able to execute simultaneous multiple requests given by various scientists. The operations manager should be able to manage the operations easily. Finally, the scientist should have easy access to the results.

The authors designed a model called DEDIP based on middle-ware architecture extending & generalizing the work of [18] and [66].

The DEDIP model has not only extended the work of [18] in multiple dimension but also generalized it to cater to all the above requirements. The DPSCH is an application where as DEDIP is an environment.

The Data product scheduler (DPSCH) proposed in [18] schedules five predefined processes. The DEDIP generalizes it to support multiple process scheduling. The process names and resources can be configured, added and altered any time.

The DPSCH schedules the processes in a fixed sequential manner. The DEDIP generalizes it to schedule processes as per the user’s choice. The user can configure it in sequential, parallel or hybrid scheduling pattern. The DPSCH has hard coded the
Overview

process scheduling. The DEDIP reads the process scheduling pattern from the user’s configuration file. The configuration file can be altered as and when required before initiating the application.

The DPSCH schedules processes only on one VAX/VMS system where DPSCH is running. The DEDIP generalizes it to support a network of multiple heterogeneous systems. The user has to simply provide the required information in his configuration file. The DEDIP expects the user to compile his application on target node and store the executable in a predefined location suggested by the DEDIP. The DEDIP works out a uniform directory structure and naming convention. This helps user in locating his application area. Furthermore, the DPSCH does not require file transfer among the nodes as it supports only single node. The DEDIP supports the automated file transfer among the nodes. The user has to give the metadata (number of files, filename, source path, destination path, etc) in a predefined format. The DEDIP transfers all the required files (ASCII or binary) from source to destination node before scheduling the dependent process. The data transfer metadata may be static for some applications and dynamic for other application. The user can manually provide the static information in the predefined file in predefined format. Some applications need to generate metadata dynamically during the processing. DEDIP supports routines (in its message-passing library) that help user in creating the dynamic information within his application. User has to simply call the routine with the required information. The routine will generate the required file in predefined format.

The DPSCH schedules five processes of only one application (Data Products generation). DEDIP supports multiple simultaneous applications. Different application designers can independently configure their applications. Operator can execute the applications as and when required. Operator can execute multiple applications simultaneously from the single terminal. The DEDIP will not only execute the tasks configured by each application but also provides various status displays to operator to monitor and control his session.

The DPSCH simulates the pipeline processing for better utilization of FPUs in data products generation. DEDIP extends the same for any resource required by an application. User configures the synchronization point for his application. DEDIP has developed resource allocation and releasing library routines. It allocates the required resource before task scheduling. The resource allocation routine puts the tasks in wait mode, incase the resource is occupied by another task. It releases the resource after completion. It was observed during the operation that some tasks need the resource only for a short duration. In such cases, application designer is advised to use the library for allocating and releasing the resource. The users are familiar to such programming as they are allocating and releasing the tape units in this mode since earlier days.
Overview

The DPSCH supports error handling for abnormal termination of any process. DEDIP extends the same to support the application level implications. If any error occurred in any process of an application, DEDIP provides the facility to restart the process, restart the entire application, suspend/resume the application or abort the application. It handles all the tasks (on host and on remote nodes) accordingly.

The DPSCH provides the SMG (Screen Management Graphics) based operator interaction on VAX/VMS. The DEDIP supports GUI (Graphical User Interface) on all the platforms. The initial versions of DEDIP started with enhancing the interaction capabilities on VAX/VMS. Then the user friendly GUI was developed on Unix, using Motif. The guideline given by Alan Cooper was followed during the GUI development [70].

The network daemon developed by [66] supports utilization of a display-device on homogeneous system configuration. The DEDIP extends the same for heterogeneous system configuration.

The model makes development of distributed image processing applications very easy. The user has to develop an application as collection of small executables (processes) instead of a big executable. These executables are interfaced through intermediate files. The scientists are very much familiar as well as acquainted with this mode of development. Then, he will have to simply configure his application package using the DEDIP graphical user interface. He has to provide the resource requirements of each process, node on which a process is to be scheduled, and the interdependency of all processes. The model schedules the processes on the target nodes as per the configuration. It provides facilities like guaranteed automatic data transfer (as per the information given in the configuration), error handling, and suspending/resuming a process or application (package). We use application and package interchangeably.

The final DEDIP model supports the following advance features:

i) Application Configuration

The DEDIP model supports a GUI that helps the application designer to provide the most crucial process interdependency information. It drives the user for providing the resource information required by each process. Please see screen shots given in chapter-7 and 8.

The DEDIP stores all the crucial information for all the applications (configured by different designers) at a centralized place.
Overview

ii) Application building

All the processes of an application are required to be built on the target node. The scheduler system carries out this activity for each process configured within an application. It copies the source code on the target node, builds the executable, stores the executable in predefined path and deletes the source code. It provides the compilation errors on the users GUI.

iii) Application execution and monitoring

It provides a smooth operational environment for executing the application. It helps the operator in navigating to the required applications and executing the same. The DEDIP schedules all the processes as per the interdependency information on the target node. It ensures that the dependent process is scheduled only when the required processes are successfully completed. DEDIP transfers the required files from one node to another node before executing the dependent process. It provides the integrated status displays for monitoring the application progress. It provides the interface for an application error recovery. The operator can suspend, resume, restart or abort an application.

An operator can start multiple applications simultaneously. DEDIP can support multiple operators too.

iv) Error Handling

In case of abnormal completion, the DEDIP 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 entire application or to abort the entire application.

DEDIP maintains complete information about the process termination. This enables the operator to carry out the error handling during the next logon also.

v) 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 background processing is very effective in the case of non-interactive processes. The DEDIP gives the detailed status to the operator at the next logon.

vi) DEDIP system management

The DEDIP system consists of a DEDIP server, DEDIP backup server and DEDIP agents. DEDIP server is the main component and hence its backup is designed. Furthermore, DEDIP server is capable of restoring its state with minimum loss of processing power in case of emergency shutdown.

The following facilities are supported in the model:

a) 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 the image-processing designer inserts the IO dependency information 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-2].

b) Pipeline processing

The DEDIP also supports the pipeline processing configuration for SAMD (Single Algorithm Multiple Data) mode. The operator can submit multiple requests for an application with different data sets in pipeline mode. The application designer (or operation manager) has to simply provide necessary synchronization information in the configuration file.
Overview

A prototype of DEDIP, using Master-Slave model, was developed [IV]. It provides a user-friendly image-processing operational environment. The system configuration consists of VAX/VMS based host and UNIX/XENIX/SOLARIS based PC/Work stations (WS). The DEDIP was tested using three simulated packages (5 test cases) by 3 operators in 10 runs. The results obtained were giving 90-95% efficiency compared to the best theoretical expectation for parallel applications. It provided a much higher efficiency (223% for parallel application to 106% for sequential application) in comparison with the task carried out manually. Intermediate results were functions of operator skill and facilities available to the operator. The prototype was developed using Fortran and ‘C’. GUI development on VAX/VMS was quite challenging due to the limited facilities available in the SMG (Screen Management Graphics) library.

Then, serious efforts were made to develop a full-fledged environment [V]. Fifteen scientists used the environment for developing & operationalizing 10 applications. The system was tested extensively and operationally used for the ten application packages. The model was further augmented to provide a facility to use the Unix system as host. Master scheduler, a core component of DEDIP, was developed on Unix. A more user friendly GUI was supported for application configuration, navigation, execution and monitoring using Motif.

A survey was conducted to quantify the usefulness of DEDIP. Fifteen key-designers were interviewed. Everyone expressed the view that DEDIP is highly required. Furthermore, they were asked to estimate the additional efforts required to develop and operationalize their distributed application in the absence of DEDIP.

- Two computer engineers demanded additional six man months per application.
- Three scientists, having 6 years of experience in developing image-processing applications, demanded specialized training and additional 11 man months per application.
- Two project managers demanded dedicated software professionals for the same. ("The additional time and training requirement will be given by the software professionals", they added).
- Eight persons were reluctant to put such complex/tedious efforts. They demanded the DEDIP or an equivalent tool as a mandatory requirement.

Please refer chapter-7 for more detail.

The next step for improving the environment was to support the other platforms to make it more system independent. The use of the Java [67] helped in making DEDIP system independent. The final model called WebDedip [VI,VII], developed in 1998-
Overview

2000, supported the three-tier architecture using Java and web technology. The model consisted of a DEDIP server running in parallel with any standard web server. The entire GUI is supported on the browser providing the roaming profile to the users. Java agents are running on various systems helping DEDIP server to provide the distributed parallel-processing environment. 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, it makes connection with DedipServer on predefined port and volunteers itself for computation workload. Java object persistence is used in storing the information, including dynamic information.

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

The system supports value added facilities for server redundancy making the model fault-tolerant.

Please refer chapter-8 for more detail.

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, meetings management, project task management, personal tasks management, document authentication, resources 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 specific functionality. Web based application development needs the client-server modeling. The DEDIP server was found to be useful for making the development easier. The DedipServer was customized to support their distributed computing requirements. These web-based applications did not require the DEDIP GUI, instead they needed direct interface with DEDIP server. The WebDedip is having its own library for communication amongst DEDIP server, DEDIP agents and DEDIP GUI. The library exploit the Java capability for object based communication. The application designers were asked to use a class named "RequestObject". They need to call only one function "RequestToServer(Object)" to interface with server. The object (passed as an argument) needs 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.
Overview

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; (1) to work out communication protocol, (2) servlet to servlet interface, and (3) 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 changed whenever a new functionality is added in an application. The DEDIP made their development very easy making the functional components independent of network communication.

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

The civil engineering people from Nirma Institute of Technology need the parallel-processing environment to conduct the research work. The collaboration is being worked for using WebDedip for the same. A simple application is implemented using DEDIP [VIII].

6. Conclusion

The optimization study for virtual parallel processing on VAX/VMS gave 300% speed-up for radiometric correction software. It also generated useful guidelines, using which other software optimizations were also able to get a similar efficiency. The ACCESS was successfully implemented on transputer network satisfying the real time requirements of IRS. This study also worked out basic guidelines that are very useful for parallel image processing. The tools developed as a part of the research work were found very useful. The DBT was found useful in a parallel processing system. The DEDIP was found extremely useful for distributed parallel processing for a network of heterogeneous system configuration. These tools are already being used by many scientists. The WebDedip is not only useful for image processing but also used for development of web applications too. Its usage in the field of civil engineering also is under study.