Chapter 9

Conclusion
Conclusion

The object of this research work was to study the user-independent optimization aspects for parallel and distributed processing.

We started our study with virtual parallel on VAX/VMS. As a result of this work, the radiometric correction software was optimized from 10 minutes to 2.75 minutes for an image [I] which is much beyond application designer. It also generated useful guidelines, using which others were able to get similar efficiency for the geometric correction and photo processing applications [I]. We could quantify not only the total improvement but also the efficiency gain for each of the guidelines.

We then extended one study for parallel machine. We studied the work carried out by various scientists in the field of image processing. We arrived at a conclusion that the systematic implementation of an application is required to study the different issues like topology selection, buffer length selection, parallelization of communication & computation, real time processing, implementation efforts v/s efficiency.

We selected “Automatic cloud cover estimation software” (ACCESS) which was demanding a real time environment. The transputer was taken as test bench as it was the most popular parallel machine when study began. We implemented parallel ACCESS using ring and tree topologies [II]. Following conclusions were drawn from the experiment:

1. The development time was almost 3 fold even after having knowledge of parallel processing and expertise in C.

2. 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.

3. 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 \( \frac{1}{4} \), to meet the requirements. We also verified that such sub sampling does not affect the accuracy of the algorithm.

4. The OCCAM is 200% efficient than the parallel C.
Conclusion

5. Almost linear scalability was observed in case of tree topology. The same was not observed in case of ring topology due to inter process communication overheads.

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

7. Buffer length of 32 KB was found the best for ACCESS.

We also concluded from literature survey, discussion with scientists and ACCESS implementation effort that an automatic parallelizing tool will be very much useful. Thereafter a study was conducted to evaluate the available automatic parallelizing tools. Most of the work being carried out was concentrated in the field of parallel compilers. The fine-grain and mid-grain parallelism was adopted for the data partitioning parallel compilers. The image processing algorithms fall into the coarse-grain parallelism in which the huge data itself is the source of inherent parallelism.

Hence, we carried a successful feasibility study developing a directive-based translator (DBT) for image processing having single partitionable input image [III]. The ACCESS was taken as a test case to prove the model. The DBT generated output results are almost matching with that of hand coded version. Thus, the DBT efficiency was found to be 92-95% compared to hand-coded version. Hence, we can conclude that the application designer can save 200% efforts at the cost of 5-8% less efficiency. Furthermore, the user need not learn parallel processing concepts, programming techniques and tedious & cumbersome debugging procedures.

The distributed processing was replacing the parallel processing in most of the fields. We conducted a feasibility study for developing DEDIP (Development Environment for Distributed Image Processing) with highly encouraging results [IV]. The focus of the study was to provide an environment using which a user can easily develop distributed parallel application. The environment should be such that the user need not bother about the complexity involved in using distributed resources, network programming using TCP/IP, data transfer, process scheduling and operation ease. All these should be the supported by the environment.

DEDIP was developed using Master-slave mode. 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 is tested using three simulated packages (5 test cases) by 3 operators in 10 runs. The result obtained was 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
Conclusion

carried out manually. Intermediate results were functions of operator skill and facilities available to the operator.

We then extended this model by developing the 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 [V]. 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 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 scientist, 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 equivalent tool as a mandatory requirement.

If we consider average 10 man months additional efforts for each application, the DEDIP saved 100 man months efforts for IRS-1C. Furthermore, it will reduce such effort for other satellites too.

The next step for the environment was to support the other platforms to make it more system independent. The final model, developed in 1998-2000, supported the three-tier architecture using the Java and web technology. The model consisted of a DEDIP server running in parallel with the any standard web server. The entire GUI is supported on the browser providing the application designer and operator a roaming profile. The Java agents are running on various systems helping DEDIP server to provide the distributed parallel-processing environment [VI, VII]. The system supports value added facilities for server redundancy. The model became truly system independent as it can be executed on any machine having Java virtual machine.
Conclusion

Thus, the optimization study for virtual parallel processing on VAX/VMS gave the 300% speed-up for radiometric correction software. It also generated useful guidelines, using which others 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 over the heterogeneous system configuration. These tools are already being used by many scientists.

Currently the usage of DEDIP in the field of Civil engineering research work is being studied[VIII]. We are confident about the success. Adaptability and load balancing are considered as extension of this study.