Design and Implementation of Parallel Execution for Scientific Computing

Developed by

Saira Junaid

Sana Zubair Khan

Supervised by

Dr. S. Tauseef-ur-Rehman

Mrs. Saima Iqbal

Department of Computer Sciences

International Islamic University, Islamabad.

2005

13-7-2010

T.2995

DATA ENTERED

@E

30/4/2012

BS

004.36

KHD

Electronic data processing—
Distributed processing.
Parallel processing
(Electronic computers)

Department of Computer Science

International Islamic University Islamabad

FINAL APPROVAL

Dated: 25-10-05

It is certified that we have read the project report submitted by Ms. Sana Zubair Khan Registration No.728-CS/BCS/00 and Ms. Saira Junaid Registration No.715-CS/BCS 00, and it is our judgment that this project is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad for the Bachelors Degree in Computer Science.

COMMITTEE

External Examiner

Mr. Shaftab Ahmed,
Senior Prinicipal Engineer,
Bahria University,
Islamabad, Pakistan

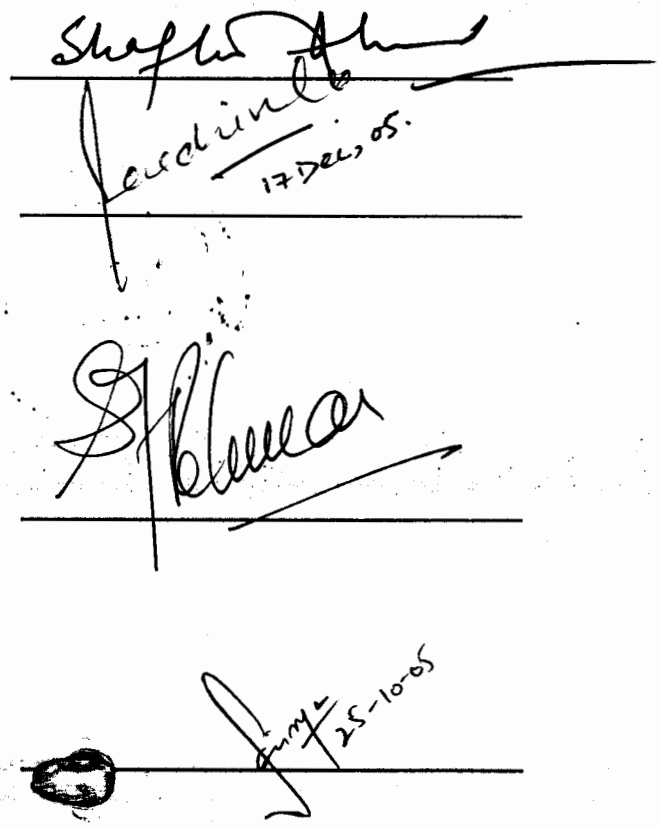
Internal Examiner

Ms. Sadia Nisar,
Research Associate,
International Islamic University,
Islamabad, Pakistan

Supervisors

Dr. S. Tauseef-ur-Rehman,
Assistant Professor,
Faculty of Applied Sciences,
International Islamic University,
Islamabad, Pakistan.

Mrs. Saima Iqbal,
Lecturer,
Faculty of Applied Sciences,
International Islamic University,
Islamabad, Pakistan.



This project is lovingly dedicated to

M.A. Jinnah

Whom we are highly indebted to

A dissertation submitted to the
Department of Computer Science,
International Islamic University, Islamabad
as a partial fulfillment of the requirements
for the award of the degree of
Bachelors of Computer Science

Declaration

We, Saira Junaid D/O Malik Junaid Akhtar and Sana Zubair Khan D/O Dr. M. Zubair Khan, hereby declare and affirm that this software neither as a whole nor as a part thereof has been copied out from any source. It is further declared that we have developed this software and accompanied report entirely on the basis of our personal efforts, made under the sincere guidance of our teachers. If any part of this project is proven to be copied out or found to be a reproduction of some other, we shall stand by the consequences.

No portion of the work presented in this report has been submitted in support of an application for other degree or qualification of this or any other University or Institute of Learning.

Saira Junaid (715-CS/BCS/00)

Sana Zubair Khan (728-CS/BCS/00)

Acknowledgement

Many thanks to Almighty Allah, the most Merciful and Compassionate, who enabled us to fulfill the task assigned to us. We would like to express our heartiest gratitude to our supervisors **Dr. S. Tauseef-ur-Rehman** and **Mrs. Saima Iqbal** for their concern and assistance all the way through this project.

Last but not the least; we would like to acknowledge the support of our family. We would like to admit that we owe all our achievements to our truly sincere and most loving parents, brothers, sisters and friends who mean the most to us, and whose prayers have always been a source of determination and motivation for us.

Saira Junaid

Sana Zubair Khan

Project In Brief

Project Title:	Design and Implementation of Parallel Execution for Scientific Computing.
Undertaken By:	Saira Junaid (715-CS/BCS/00) Sana Zubair Khan (728-CS/BCS/00)
Supervised By:	Dr. S. Tauseef-ur-Rehman Mrs. Saima Iqbal
Starting Date:	October 10, 2003
Completion Date:	March 14, 2004
Tools Used:	MPICH -1.2.5.2 JBuilder 8.0
Operating System:	Linux Redhat 8.0
System Used:	Pentium IV

ABSTRACT

First part of the thesis contains the brief introduction to Parallel Computing as well as Cluster Technology. Then the scope of the software and objectives of the project have been discussed.

Next chapter describes the existing scenario and the problems that were encountered during the detail study of the implementation and development of clusters in Linux environment.

In the next chapter, the proposed system is discussed in detail emphasizing on its feasibility and features. A detailed system analysis is done in order to proceed towards system design. Afterwards, the independent modules of the system have been decomposed following the proper steps leading towards the near-to-implementation stage in the product life cycle.

As the platform of cluster technologies is new, a detailed account has been added to introduce the yet new technology. Parallel Computation platform is discussed along with the use of Linux and Cluster technology.

The product is then fully tested and the results are placed in the chapter related to testing. In the end, the benefits and limitations of the project have been fully discussed and definitions of terms unknown to the common man have been provided.

Saira Junaid
Sana Zubair Khan

Table of Contents

Chapter No. Contents **Page No.**

CHAPTER 1

INTRODUCTION

1 Introduction

This chapter describes the introduction of Parallel and Cluster Computing, the scope, vision and the main features of the project.

1.1 Introduction to Project

The concept dates back to the 1970's when IBM implemented some aspects of clustering in its mainframe products. The first real cluster product appeared in 1982 when the Digital Equipment Company (DEC) introduced its VAXCluster. The VAXCluster offered more economical computing by uncoupling the Input/Output (I/O) devices from any single CPU. Instead, all CPUs could access the devices and their contents via a star topology bus and coupling devices.

From this rather simple beginning, clustering has developed into a complex branch of parallel computing. Numerous factors contribute to this. First, there is no standard for clustering computers. Clusters can be implemented in many different business problems using numerous topologies. In addition, there is no standard platform to build a cluster. Uni-processor and multiprocessor machines from all vendors can be mixed and matched in clusters as well. Clusters can be built using Reduced Instruction Set Computing (RISC), Complex Instruction Set Computing (CISC) or even Very Large Instruction Word (VLIW) processors.

For many businesses, computing systems have become such an integral part, that severe consequences can occur when they are unavailable. These systems are used for critical business functions such as order processing and tracking, inventory control, transaction processing, customer support and e-commerce and when these critical functions are not accessible it can result in lost revenue, lost productivity, reduced customer satisfaction, possible data loss or reduced decision making capabilities. As a result, in today's global environment, these computing systems must be available 24 hours a day round the year. As the performance of commodity computer and network hardware increases, and their prices decrease, it becomes more and more practical to build parallel computational systems from off-the-shelf components, rather than buying very expensive Supercomputers. In fact, the price per performance ratio is between three to ten times better than that of traditional supercomputers.

1.2 Area of Knowledge

This part of the chapter describes the areas of interest for the project, which are described as follows:

1.2.1 Parallel Computing

Parallel Processing refers to the concept of speeding-up the execution of a program by dividing the program into multiple fragments that can execute simultaneously, each on its own processor. Ideally, a program being executed across *n* processors might execute *n* times faster than it would use a single processor; however, in practice we are bound by Amdahl's Law.

Traditionally, multiple processors were provided within a specially designed "parallel computer"; along these lines, Linux now supports **SMP** systems (often sold as "servers") in which multiple processors share a single memory and bus interface within a single computer. It is also possible for a group of computers (for example, a group of PCs each running Linux) to be interconnected by a network to form a parallel-processing **cluster**. The third alternative for parallel computing using Linux is to use the **multimedia instruction extensions** (i.e., MMX) to operate in parallel on vectors of integer data. Finally, it is also possible to use a Linux system as a "host" for a specialized **attached** parallel processing compute engine. All these approaches are discussed in detail in this document.

By means of Parallel Programming, we can split a program in multiple tasks running on different PCs, which exchange data in cooperation. So we can take advantage of memory and calculation power of many PCs in parallel, while considerably decreasing the time of program execution. Parallel computing extends to systems with more processors to obtain speedup in code execution. The efficiency and effectiveness of the parallelism are largely dependent on the problems to be solved with selected techniques and hardware architectures. Scientific applications are the major motivation behind the development of parallel computing.

1.2.2 Cluster Computing

Cluster is a loosely coupled set of computers that function as a single computer thus; it qualifies as a branch of parallel computing. Specifically, clusters are a distributed form of parallel computing. Implementations and topologies of clustering vary significantly with the degree of parallelism and the function, physical platform, operating system, network, and so on.

A cluster is made-up of a group of personal computers interconnected by a fast network. The cluster nodes have neither monitor, nor keyboard, but they pass on additional computing power as well as memory. The first issue is the synchronization of the operation system and for this; we have to modify the code. Fortunately, many commercial vendors supply this functionality or provide APIs to build such a capability in an operation system.

Clustering is most widely recognized, as the ability to combine multiple systems in such a way that they provide services a single system could not. Clustering is used to achieve higher availability, scalability and easier management. Higher availability can be achieved by use of

fail-over cluster, in which resources can automatically move between two or more nodes in the event of a failure. Scalability can be achieved by balancing the load of an application across several computer systems. Similar management can be achieved through the use of virtual servers, as opposed to managing each individual system.

The power of cluster is due to the collaboration of many computing units in order to solve a complex problem, by dividing it in many tasks appropriately assigned to each PC. To make the simulation consistent, the cluster nodes can exchange data during the computation. These features highlight how clusters are based on an intelligent collaboration philosophy, which makes available a remarkable power for scientific computations.

1.3 Need of Project

Parallel computers are more difficult to program than computers with a single processor because the architecture of parallel computers varies widely and the work of multiple processors must be coordinated and synchronized. Since the processors depend on some type of communication network to share data, several models for connecting processors and memory modules exist, and each topology requires a different programming model. The three models that are most commonly used in building parallel computers include synchronous processors each with its own memory (also known as distributed memory), asynchronous processors each with its own memory, and asynchronous processors with a common, shared memory.

As the hardware for the parallel computers is very expensive and difficult to find and it is also very difficult to program in parallel computers, clusters can be used for the same purpose and they also speed-up many operations. Because many organizations lack the funds to purchase expensive parallel computers, cost effective alternatives are needed for parallel processing. The computers based on the above mentioned three models can be served to fulfill the demands of parallel computing. A small cluster can be made by joining a number of processors, to work effectively and in parallel, to solve complex computational problems, reducing the time consumed by the single processor. Parallel processing using Linux can yield efficient performance for some programs that perform complex computations or operate on large data sets. What is more, it can do that using cheap hardware which we might already own. As an added bonus, it is also easy to use a parallel Linux system for other things when it is not busy executing a parallel job.

1.4 The Scope and Vision of the Project

The future of parallel computing is promising. As new parallel technologies are developed to address the need for high performance computing, they are making their way into the mainstream of computer science education. The demand for parallel computing is expected to grow, and many organizations are planning to introduce themselves to parallel processing and are seeking resources to do so. Clusters are currently both the most popular and the most varied approach, ranging from a conventional network of workstations (NOW) to essentially custom parallel machines that just happen to use Linux PCs as processor nodes. There is also quite a lot of software support for parallel processing using clusters of Linux machines.

During the past 10 years, the trends indicated by ever-faster networks, distributed systems, and multi-processor computer architectures (even at the desktop level) suggest that parallelism is the future of computing.

1.5 Project Description

The goal of this project is to pursue the development and deployment of High Performance Computing using Linux Clusters. The purpose of this project is to develop a diagnostic program that can be executed on a Cluster to verify functionality and certify compatibility. The software provides a simple mechanism for novice users to test a cluster. In addition, its modular and extensible design allows users to easily customize the software to fit their needs.

1.6 Major Modules of Project

There are two major modules in the project including a Cluster Management System and an Integrated Development Environment. These are as follows:

1.6.1 Cluster Management System

Cluster Management System is a collection of fully integrated, easy to install software components designed to make it easy to build and use a cluster for high performance computing. Everything we need to build, maintain, and use a modest sized cluster has been included. In other words CMS contains the resources we need to apply cluster computing to our High Performance Computing problems.

This software is designed to administer and manage application jobs submitted to workstation clusters. It encompasses the traditional batch and queuing systems. The main reason for their existence is their ability to provide an increased and reliable throughput of user applications on the systems they manage.

The CMS works completely outside the kernel and on top of a machine's existing operating system. This means that its installation does not require modification of the kernel, and so basically the CMS package is installed like any other software package on the machine.

Cluster Management Software has been primarily designed to administer and manage application jobs submitted to a cluster and to manage other underlying resources, such as memory load on each node, memory consumption by a single program etc.

1.6.2 Integrated Development Environment

In our project an integrated development environment (IDE) comprising of a text editor and compiler has been developed. This IDE is devoted to a specific programming language in case of our project i.e. C language. It enables development and execution of Cluster Aware Applications.

1.7 Objectives of the Project

The following are the key objectives to be achieved through the life cycle of the project:

1. The field of cluster computing is relatively new, with advancements being made now and then. Therefore in order to keep Pakistan in pace with the changing world of technology, this project was taken.

2. To introduce a base line for projects requiring massive computations like War Games, Nuclear and Space Projects etc.

3. To provide CMS as an alternative to Symmetric Multiprocessors (SMPs) as they are more economical as well as technically vulnerable to user requirements as compared to SMPs.

4. To provide an economical alternative for super computers/ mainframes.

1.8 Tools & Languages Used

- MPICH 1.2.5.2
- JBuilder 8 Enterprise Edition
- Java 2 Standard Edition (J2SE)
- Borland C++
- Linux Red hat 8.0

CHAPTER 2

EXISTING SYSTEM

2 Existing System

Due to rapid development in the field of parallel computing, different libraries are being developed which support programming for different parallel architectures. Different compilers and APIs are used according to different architectures. Scalable parallel computing on PC clusters requires the use of a message passing system such as MPI (Message Passing Interface) or PVM (Parallel Virtual Machine), although OpenMP and other forms of thread-based parallelism may also be used on SMP nodes. Programming languages of interest for scientific computing on PC clusters include Fortran, C, and C++, although Java is also being used increasingly.

2.1 Existing Parallel Models

Using the power of parallelism within application programs to achieve maximum performance has been a challenge for high end computing. Vector supercomputers, SIMD array processors, and MPP multiprocessors have used varying forms of algorithmic parallelism. The results have been mixed. The highest degrees of performance yet achieved have been through parallel computation. But in many cases, the efficiencies observed have been low and the difficulties and costs involved in their accomplishment have been high. Among many multiple programming models that have been used, two have emerged as better choices than others. These are "communicating sequential processes" model usually referred to as the "message passing" model and the "Threading" model.

2.1.1 Threading Model

Threads are a popular paradigm of parallel programming on uni-processor systems as well as multi-processor systems. On multi-processor systems, threads are used to simultaneously utilize all the available processors. In uni-processor systems, threads are used to utilize the resources effectively. This is done by using the asynchronous behavior of an application for overlapping computation and communication. Multi-threaded applications respond more quickly to the user inputs and run faster. Threads communicate using shared variables created within their parent process address space.

2.1.2 Message Passing Model

Message passing allows efficient parallel programs to be written for distributed memory systems. These libraries provide routines to initiate and configure the messaging environment as well as sending and receiving packets of data. Currently the most popular message passing system, which has evolved as a standard for parallel systems, especially cluster (clusters are distributed memory parallel systems) is the "MPI" (Message Passing Interface standard). MPI is now found on virtually every multiprocessor system including SMPs, MPPs, and clusters. MPI is not a full language but a library that allows users of Java, C/C++ and FORTRAN to access libraries for passing messages between concurrent processes on separate but interconnected processor. A number of implementations of MPI and many open source versions are available. Along with these libraries, some tools and programming

environment is also required to understand the operation of a program for its debugging and to enhance performance. Such tools for clusters are in their initial stages of development, although significant efforts are being made all over the world but no true widely accepted standard has yet emerged.

2.2 Commonly Used Message Passing Systems

There are many massage passing systems being used nowadays but the most widely used are:

1. Parallel Virtual Machine (PVM)
2. Message Passing Interface (MPI)

2.2.1 Parallel Virtual Machine (PVM)

PVM is an integrated set of software tools and libraries that emulates a general-purpose, flexible, heterogeneous concurrent computing framework on interconnected computers of varied architecture. The overall objective of the PVM system is to enable such a collection of computers to be used cooperatively for concurrent or parallel computation. The principles upon which PVM is based include the following:

i) User-configured host pool: A PVM application runs on a set of machines that are selected by the user. Both shared and distributed-memory computers may be part of the host pool.

ii) Definable access to hardware: A PVM application views the hardware environment as an attribute-less collection of virtual processing elements or user specified to take advantage of different machine capabilities.

iii) Process-based computation: The unit of parallelism in PVM is an independent sequential thread of control that alternates between communication and computation. Multiple tasks may execute on a single processor.

iv) Explicit message-passing model: Collections of computational tasks, each performing a part of an application's workload using data, functional, or hybrid decomposition, cooperate by explicitly sending and receiving messages to one another.

v) Heterogeneity support: The PVM system supports heterogeneity in terms of machines, networks, and applications.

vi) Multiprocessor support: PVM uses the native message-passing facilities on multiprocessors to take advantage of the underlying hardware. Vendors often supply their own optimized PVM for their systems, which can still communicate with the public PVM version.

The PVM system is composed of two parts. The first part is a daemon, which resides on all the computers making up the virtual machine. The daemon is designed so any user with a

valid login can install this daemon on a machine. When a user wishes to run a PVM application, he/she first creates a virtual machine by starting up PVM. The PVM application can then be started from a UNIX prompt on any of the hosts. Multiple users can configure overlapping virtual machines, and each user can execute several PVM applications simultaneously.

The second part of the system is a library of PVM interface routines. It contains a functionally complete collection of primitives that are needed for cooperation between tasks of an application. This library contains user-callable routines for message passing, spawning processes, coordinating tasks, and modifying the virtual machine.

2.2.2 Message Passing Interface (MPI)

This message-passing model of parallel computation has emerged as an expressive, efficient, and well-understood paradigm for parallel programming. Until recently, the syntax and precise semantics of each message-passing library implementation were different from the others, although many of the general semantics were similar. The proliferation of message-passing library designs from both vendors and users was appropriate for a while, but eventually it was seen that enough consensus on requirements and general semantics for message-passing had been reached that an attempt at standardization might usefully be undertaken.

The process of creating a standard to enable portability of message-passing applications codes began at a workshop on Message Passing Standardization in April 1992, and the Message Passing Interface (MPI) Forum organized itself at the Supercomputing '92 Conference. During the next eighteen months Version 1.0 of the MPI Standard was completed. Important contributions came from Zipcode, Chimp, PVM, Chameleon, and PICL.

The features of MPI are as follows:

i) MPI includes point-to-point message passing and collective operations, all scoped to a user-specified group of processes. Communicators, which house groups and communication context information, provide an important measure of safety that is necessary and useful for building up library-oriented parallel code.

ii) MPI also provides three additional classes of services: environmental inquiry, basic timing information for application performance measurement, and a profiling interface for external performance monitoring.

iii) MPI makes heterogeneous data conversion a transparent part of its services by requiring data-type specification for all communication operations. Both built-in and user-defined data-types are provided.

iv) MPI accomplishes its functionality with opaque objects, with well-defined constructors and destructors, giving MPI an object-based look and feel. Opaque objects include groups, communicators, and request objects for asynchronous operations. User-defined and predefined data types allow for heterogeneous communication and elegant description of gather/scatter semantics in send/receive operations as well as in collective operations.

v) MPI provides support for both the SPMD and MPMD modes of parallel programming. Furthermore, MPI can support inter-application computations through inter-communicator operations, which support communication between groups rather than within a single group. Dataflow-style computations also can be constructed from inter-communicators.

vi) MPI provides a thread-safe application-programming interface (API), which will be useful in multithreaded environments as implementations mature and support thread safety themselves.

Our Integrated Development Environment (IDE) is designed for writing, compiling and running the Cluster Aware Applications (CAAs) in C. The IDE uses MPICH implementation of MPI (Message Passing Interface) support for C for developing the CAAs. The reason for using MPI is that MPI and Parallel Virtual Machine (PVM) are the only APIs used currently for CAA development. MPI was chosen because the IEEE standard API for parallel/cluster computing has approved it. Therefore, there was no need to develop custom API for the purpose of message passing.

CHAPTER 3

SYSTEM ANALYSIS

3 System Analysis

The synthesis of a target system model is a transgression into the computational realm of design. It cannot be denied that constructing a model of an intended system has the flavor of design, at least to the extent that some commitments are made with respect to (logical) system architecture. At the same time, the relative concreteness of a model is an advantage for all parties involved. Analysts are forced to think through the demands of the customer from yet another perspective. The analysts and customers through mentally executing the scenarios can check the validity of a model. Designers will obtain an abstract model that may be transformed into an executable realization.

3.1 Object-Oriented Analysis Method

The OO analysis method consists of three parts: Process, Modeling Language and a CASE tool. The process serves as the project roadmap that defines who (roles) is doing what (activities performed and artifacts produced), when (time order of activities) and how (guidelines and templates).

As visual languages are often technology driven hence for the requirements and system analysis, three approaches are normally used depending on the type of development strategy used for the software and the area to which it is related. In case of structured programming data flow diagrams (commonly known as DFD) are made whereas for Object Oriented development unified modeling language may be employed (UML). For software projects dealing with databases Entity Relationship diagrams are made for the system analysis. Although requirements capture is supposed to be independent of implementation, yet, it often uses one of the above-mentioned techniques.

One of the objectives of system analysis process is to be able to generate a collection of use cases. The idea is to be able to catalog and reference this collection, which serves as the user's view of the system. When it's time to upgrade the system, the use case catalog serves as a basis for gathering the requirements of the upgrade.

As this project is related to parallel computing, each of which has been developed using the object oriented concepts and technologies, hence the system analysis was done by employing unified modeling language by making use cases of all possible scenarios for each module.

3.1.1 Initial System Study

The system study is important as by understanding the existing system, we can propose an efficient solution that will fulfill the requirements and will also be able to solve the problems faced by the organization.

In initial study phase, the main objective is to decide whether the proposed solution is feasible or not i.e. it fulfills all the requirements of the user more efficiently, reliably and at a low cost. Generally, initial system study consists of following types of study

1. Financial Feasibility
2. Technical Feasibility

1. Financial Feasibility

In this phase, the benefits of the software being developed are compared with the cost of development. If the new developed software is more efficient, reliable, accurate, and easy to use than the existing system and its benefits justify the cost of development, then this project is considered financially feasible for the user.

As more businesses become dependent on computing as the backbone of their operation, the more critical is that the services provided are much more efficient and available round the clock.

Protecting a large investment in computing is a concern anytime a new purchase is considered. The rapid pace at which technology becomes obsolete gives pause to customers who want to ensure that the systems they buy today will remain useful for as long as possible. They want to know that as their business needs grow; their computing ability can grow along with it. They want to avoid having to replace equipment that they just purchased. Keeping in view such constraints faced by the normal users clusters are the most ideal solution. In our case, we are going to implement cluster.

Although clustering is well suitable for higher availability and scalability, but the emphasis of this project is on scalability. The domain of the project is to provide a low cost solution for parallel computing using clustering or give a substitute for mainframe/super computer, which is most cost effective as compared to mainframe/super computer. It is equivalent to the performance and computational power of these computing giants.

2. Technical Feasibility

Technical Feasibility focuses on whether the technology needed for the proposed system is available to the user or organization and whether this software can be easily used or requires special training.

Cluster Management System uses hardware as well as software that are both on hand and easily available for the individuals as well as organizations. Once the software is installed, any user ranging from novice to old hand users can use them without any need of special training. Hence this project is also technically feasible.

3.2 Design Objectives of the Proposed System

The objective of object-oriented analysis is to develop a series of models that describe computer software as it works to satisfy a set of customer-defined requirements. Before designing any system, it is beneficial to establish the objectives of the proposed system along with its relative importance. The design objectives of the proposed system are as follows:

3.2.1 Scalability

The clusters have an ability to grow in overall capacity and to meet higher usage demands as needed. When an application or department needs additional computational resources, additional nodes can be added to the cluster. Many clusters continue to grow and, are now comprised of thousand nodes.

3.2.2 Availability

In a business environment, many activities are automated. However, a problem will arise if the server fails. The activities could come to halt. Such situations can cause a great deal of inconvenience and a result of loss of business. This is where clusters can be useful. An organization could continue to operate even after the failure of a server by automatically isolating failed components and migrating activities to alternative resources as a means of offering an uninterrupted service.

Removal of any single point of failure in hardware and software ensures that any one system component, the system as a whole, or the solution (i.e., multiple systems) stay highly available. There is a greater level of availability in a clustering solution as the components can be isolated and in many cases the loss of a computing node in the cluster does not have a large impact on the overall cluster solution. The workload of that node will be allocated amongst the remaining computing nodes. For instance, this increases availability in geographical disaster situations.

3.2.3 Ease of Technology Refresh

In a clustering environment, integrating new processor, memory, disk, or operating system technology, can be accomplished with relative ease. As technology moves forward, modular pieces of the solution stack can be replaced as time, budget and needs require or permit. There is no need for a one-time 'switch-over' to the latest technology.

3.2.4 Efficiency

Efficiency is defined as the economic utilization of the available resources for the achievement of predetermined objectives. The proposed system is more efficient in terms of its output and through put.

3.2.5 System Manageability

System management is the installation, configuration and monitoring of key elements of computer systems, such as hardware, operating system and applications. Most of large SMPs

have proprietary enabling technologies (custom hardware extension and software components) such that they complicate the system management. On the other hand, it is easier to manage one large system compared to hundreds of nodes. With wide deployment of network infrastructure and enterprise management software, it becomes easy to manage multiple servers of a Cluster system from a single point.

3.2.6 Performance

The use of clusters as a platform for running high-performance applications is rapidly increasing. The performance of cluster components has almost reached the performance of those used in supercomputers in addition the commodity components are improving in terms of performance and functionality all the time. The performance of workstation is doubling every 18 to 24 months. The performance of network, hardware and software is improving with ever increasing bandwidths between cluster nodes.

3.2.7 Vendor Lock-in

Proprietary solutions require a commitment to a particular vendor whereas industry-standard implementations are interchangeable. Many of the proprietary solutions require only components that have been developed by that vendor. Depending on the revision and technology, application performance may be diminished. Clusters enables solutions to be built from the best performing industry standard components.

3.2.8 Installation and Service & Support:

Specialized equipment generally requires expert installation teams trained to handle such cases. Not only that they require dedicated facilities such as power, cooling, etc. For Clusters, since the components are commodity of the shelf, installation is generic and widely supported.

Also, total cost of ownership including post-sales costs of maintaining the hardware and software, from standard upgrades to unit replacement to staff training and education is lower when compared to proprietary implementations that typically come with a high level of technical services due to their inherently complex nature and sophistication.

3.2.9 Cost Factors

The use of clusters as a platform for running high-performance and high-availability applications is mainly increasing due to their cost-effective nature. The introduction of LINUX and the Windows NT operating system has led to increasing levels of interest in utilizing PC-based systems as a cost-effective computational recourse for parallel computing. The development tools and programming environments for PCs/Workstations are becoming mature, more sophisticated, and compatible to the contrasting solutions for traditional parallel supercomputers.

3.3 Features of the Proposed System

The main features of the proposed system are as follows:

3.3.1 Cluster Management System

It is primarily designed to administer and manage application jobs submitted to a cluster, management of other underlying resources, such as memory load on each node, memory consumption by a single program etc.

The job can be a parallel or sequential application that needs to run on the cluster. CMS can be used to help manage clusters in a variety of ways:

i) Optimize the use of the available resources for parallel and sequential jobs
ii) Prioritize the usage of the available resources
iii) Manage mechanisms to "steal" CPU cycles from cluster machines
iv) Enable check-pointing and task migration
v) Provide mechanisms to ensure that tasks complete successfully.

A structural view of CMS is shown in figure 3.1.

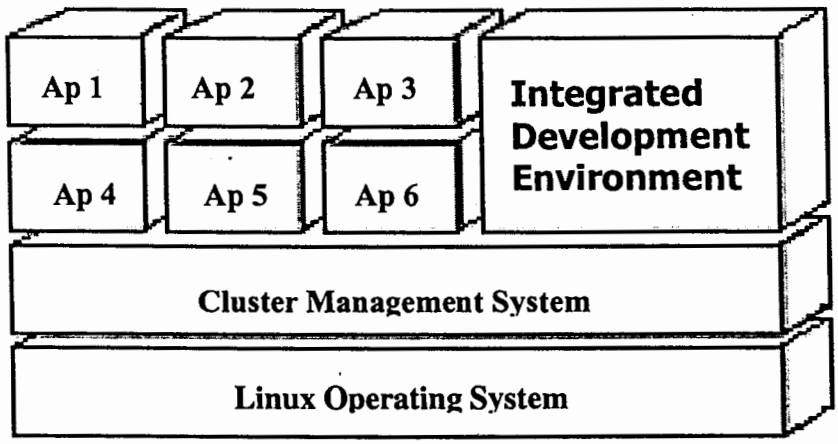


Fig. 3.1. Structural View of CMS with IDE

Figure 3.1 clearly depicts that the CMS is lying over the Operating System as a middle ware between the user applications and OS. Salient features/services provided by the CMS are:

3.3.1.1 Initialize, Shutdown, Restart Cluster

Users have an option to manually initialize the cluster by defining a server and a client node. This is the minimum requirement of a Cluster to have a server and at least 1 client node attached to the server.

The user can, at any time, shutdown the Cluster as to get rid of the serve/client architecture and to make the server node act like a standalone pc. The files are automatically updated with the information.

At any point, the users may restart the cluster. Under this option, the initial server and the client defined, in the initialize cluster dialog box, will make the cluster. If the user wants to specify some other server or the starting client, the Initialize Cluster process may be invoked.

3.3.1.2 Node Addition and Deletion

At the server, the users have an option to customize the number of nodes according to their requirement. They can have their cluster comprising of as many nodes as they need.

3.3.1.3 Open IDE

The user is given the options to open the IDE in which several programming services (editing, compiling, running, type checking, debugging etc.) are integrated under a graphical interface.

3.3.2 Integrated Development Environment

Our Integrated Development Environment (IDE) is designed for writing, compiling and running the Cluster Aware Applications (CAAs) in C. The IDE uses MPICH implementation of MPI (Message Passing Interface) support for C for developing the CAAs. The reason for using MPI is that MPI and Parallel Virtual Machine (PVM) are the only APIs used in CAAs. MPI is selected because the IEEE as standard API for parallel/cluster computing has approved it. Therefore, there is no need to develop custom API for the purpose.

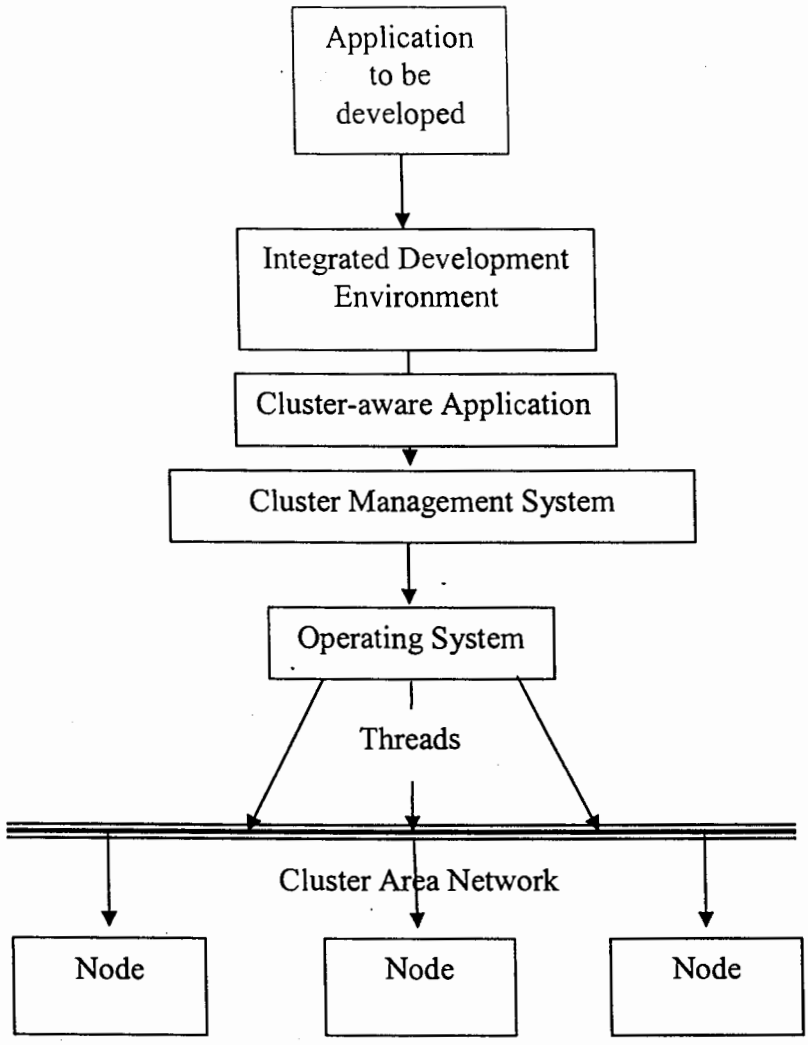


Fig. 3.2. Structural View of IDE

The main features of the IDE are:

3.3.2.1 New/Open File Options

The user can either open a new untitled file or write a Cluster Aware application or can open an already created file.

3.3.2.2 Save/Save as Options

The user can also save the file created/modified

3.3.2.3 File Edit Options

User also gets an option to cut, copy or paste.

3.3.2.4 Compile/Run File

The Cluster Aware Applications in C are compiled to see for the possible compilation errors. The file, already compiled, can be run and the output can be seen on in the output area.

3.3.2.5 Set Properties

The user can customize the number of nodes on which he wants to run the application. By default, 2 nodes are chosen.

3.4 Software Modeling

Modeling is he designing of software applications before coding. Modeling is an essential part of large software projects, and helpful to medium and even small projects as well. Using a model, those responsible for a software development project's success can assure themselves that business functionality is complete and correct, end-user needs are met, and program design supports requirements for scalability, robustness, security, extendibility, and other characteristics, before implementation in code renders changes difficult and expensive to make.

3.5 UML, A Unified Approach To OOA

Models are constructed to learn about the interesting properties of a system without constructing the actual system cost, experimenting with the actual system availability and having to view the entire system because it may be too complex to understand.

The UML consists of a number of graphical elements that combine to form diagrams. The purpose of the diagrams is to present multiple views of a system i.e. a model. An UML model describes what a system is supposed to do. It doesn't tell how to implement the system. The UML enables system builders to create blueprints that capture their visions in a standard, easy-to-understand way and communicate them to others. Conscientious system design involves all the possible viewpoints and each UML diagram gives a way of incorporating a particular view for the satisfaction of every type of stakeholder.

3.6 Use Cases, A Tool For System Analysis Using UML

A use case is a specific way of using the system by using some part of the functionality. Each use case constitutes a complete course of events initiated by an actor and it specifies the interaction that takes place between an actor and the system. A use case is thus a special sequence of related transactions performed by an actor and the system in a dialogue. The collected use cases specify all the existing ways of using the system.

The use case is a construct that helps analysts work with users to determine system usage. It is an excellent tool for stimulating potential users to talk about a system form their own viewpoints. The idea is to get system users involved in the early stages of system analysis and design. This increases the likelihood that the system ultimately becomes a boon to the people its supposed to help.

Use cases crop up in several phases of the development process. They help with the design of a system's user interface, they help developers make programming choices, and they provide the basis for testing the newly constructed system.

As powerful as the use case concept is, the use cases become even more powerful when UML is employed to visualize them. This visualization encourages the users to give you more information and helps break the ice.

3.6.1 Use Case Diagram

Following is the Use Case Diagram of the project:

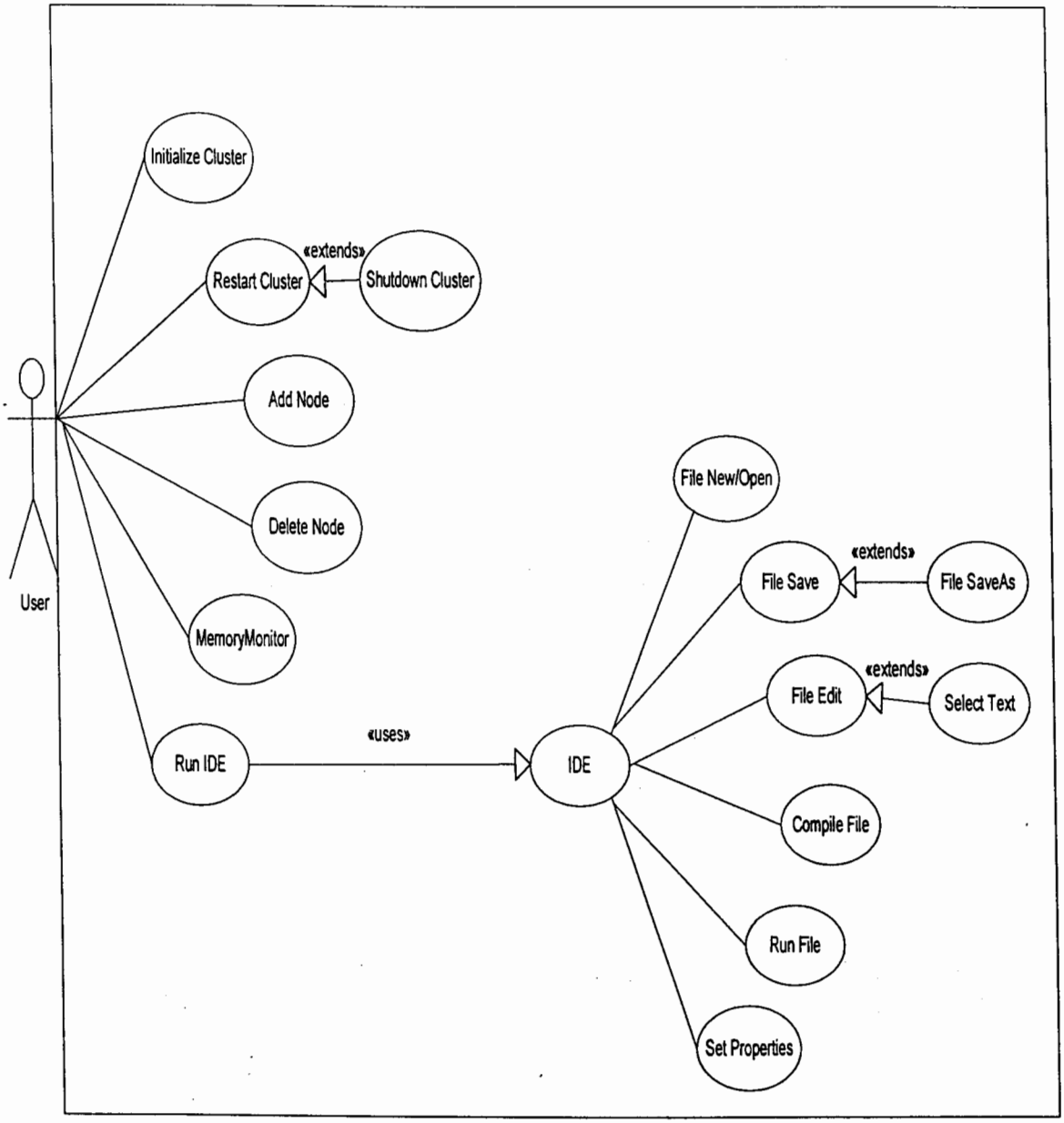


Fig. 3.3. Use Case Diagram of CMS

3.7 Use Case Description

For each module of the project several use cases were defined and the description of each use case of the CMS is as follows:

3.7.1 Use Case Description of Cluster Management System

The description of the use cases identified in the Cluster Management System is as follows:

a. Initialize Cluster

This use case initializes the cluster by initializing the server node and a single client node is asked for from the user. This use case updates the required files and gives a message that the cluster has been initialized.

b. Shutdown Cluster

This use case is only enabled after the cluster is initialized. This use case shuts down the cluster, previously initialized. The specific files are automatically updated. The user is informed that the cluster has been shutdown as an output of the use case.

c. Restart Cluster

This use case restarts the cluster. Under this option, the initial server and the client defined, in the initialize cluster dialog box, will make the cluster. If the user wants to specify some other server or the starting client, the Initialize Cluster process may be invoked. This use case is only enabled after the cluster is initialized.

d. Add Node

This use case is only enabled after the cluster is initialized. This use case takes the input from the user for adding the node in the form of the IP address and the name of the node. It checks that the node name is given as a string. If not, the values are not saved and the user is notified to enter the correct value in the fields. It also checks if the IP address entered is in the correct format. The output (that the nodes have been added) is that the added nodes are displayed in the List Box. The files are automatically updated. The user has also an option to open the Delete Node Dialog from this dialog.

e. Delete Node

This use case deletes the selected node. The already added nodes are displayed in a list box from which the user can select the node and delete it from pressing the delete button. If no node is selected, no node is deleted and the user is notified to select a node first. The user also has an option to open the Add Node Dialog from this Dialog. This use case is only enabled after the cluster is initialized.

f. Memory Monitor

This use case opens a Memory Monitor to the user. This memory monitor is a display of the memory that the processor is using at that specific time. This use case is invoked by clicking at the Memory Monitor button in a menu.

g. Run IDE

This use case opens the IDE application to users. This use case is only enabled after the cluster is initialized. This use case is invoked by clicking either the button on the front of the software of by clicking the Run IDE under the Run Application menu.

3.7.2 Use Case Description of Integrated Development Environment

The description of the use cases identified in the Integrated Development Environment is as follows:

a. File New/Open

The use case opens a new file to write a Cluster Aware Application in C. The file is names 'untitled.c' by default. This use case can be invoked by either selecting 'new' from the menu 'file' or by pressing the 'new' icon made on the tool bar.

The use case opens a file chooser to specify the file to open. This use case can be invoked by either selecting 'open' from the menu 'file' or by pressing the 'open' icon made on the tool bar.

b. File Save/SaveAs

When the user selects 'Save' from the menu displayed on choosing file from the main menu of game, this use case is activated. If the work was being done on the file 'untitled.c', the Save As dialog box will open to specify the name of the file to b saved. Else, if the user had opened an already named file and was working in it, they changes are going to get saved under the same name.

The user may select 'Save As' from the menu displayed on choosing the 'File' menu or this use case can also be invoked by pressing the 'Save As' icon. This would open the 'Save As' dialog box as to take the name and the place where the user wants to save the file.

c. File Edit

This use case contains the functions of file editing. By using this use case, any of the file edit functions like Cut, Copy and Paste maybe be invoked and used by the user.

d. Select Text

This use case, when invoked, selects all the text available on the text area. . This use case is invoked either by selecting 'Select All' from the 'edit' menu. It only works if there is text available to be selected.

e. Compile

This use case compiles the opened Cluster aware applications in C. To invoke this use case, the file has to be saved as a c source file, else it does not get compiled and an error message is displayed. Also, if the user tries to compile the unsaved file, the 'Save As' dialog box opens first and the file only gets compiled after it is saved. This use case can be invoked by pressing the 'Compile' button present on the tool bar, or by choosing 'Compile' from the 'Run' menu.

f. Run

This use case runs the opened Cluster aware applications in C. This use case first compiles the file and then runs it, in case the file has not already been run. To invoke this use case, the file has to be saved as a C source file, else it does not get compiled and an error message is displayed. Also, if the user tries to run the unsaved file, the 'Save As' dialog box opens first and the file is only run after it is saved. This use case can be invoked by pressing the 'Run' button present on the tool bar, or by choosing 'Run' from the 'Run' menu.

g. Set Properties

This use case specifies the number of nodes for the Cluster Aware Application to be run on. It opens a dialog box in which the user specifies the number of nodes he needs. By default, 2 nodes are specified. The input of the use case is in digits and it checked. Also if the user tries to specify more nodes than the cluster comprises of, an error message is displayed and the input is not taken.

3.8 Use Cases In Expanded Format

The use cases for each module are described below in the expanded format.

3.8.1 Use Cases of Cluster Management System

The use cases for the Cluster Management System are as follows:

a. Initialize Cluster

Name:	Initialize Cluster Use Case
Actor:	Cluster User
Type:	Primary, Concrete, Essential
Overview:	This use case invokes the initialization process of the Cluster
Triggering Event:	User invokes 'Initialize Cluster' from the 'Cluster Management' menu.
Precondition:	The Cluster Management Software is opened.
Post Condition:	Cluster Initialized

Success Scenario

	Actor Action		System Response
1	Click on menu 'Cluster Management' → Initialize Cluster menu		
2	Enter Client Node name and IP	3	System will Initialize the Cluster

Failure Scenario

	2a Invalid Input
	3a Display Error Message "Unable to Initialize Cluster"

b. Shutdown Cluster

Name:	Shutdown Cluster use case
Actor:	Cluster User
Type:	Primary, Concrete, Essential
Overview:	If the Cluster is already initialized, invoking this use case would Shutdown the cluster.
Triggering Event:	User invokes 'Shutdown Cluster' from the 'Cluster Management' menu.
Pre Condition:	The cluster is initialized.
Post Condition	Clusters Shuts down.

Success Scenario

	Actor Action		System Response
1	Click on menu 'Cluster Management' → Shutdown Cluster menu	2	Cluster Initialization is Validated
		3	Shutdown Cluster

Failure Scenario

	2a Display Message "Unable to Shutdown Cluster"

c. Restart Cluster

Name:	Restart Cluster use case
Actor:	Cluster User
Type:	Primary, Concrete, Essential
Overview:	After Shutting down the cluster, by invoking this use case, the user can restart the Cluster
Triggering Event:	User invokes 'Shutdown Cluster' from the 'Cluster Management' menu
Pre Condition:	The Cluster has been Shutdown
Post Condition	The Cluster gets into its working condition

Success Scenario

	Actor Action		System Response
1	Click on Cluster Management → Restart Cluster menu	2	Cluster Restarted.

Failure Scenario

	2a Display Message "Cluster Already Start"

d. Add Node

Name:	Add Node use case
Actor:	Cluster User
Type:	Primary, Concrete, Essential
Overview:	The user adds nodes to the cluster
Triggering Event:	User invokes "Add Nodes" from the 'Node Management' menu
Precondition:	The cluster is initialized
Post Condition:	The nodes are added and the files are updated.
Constraints:	The use may add one node at a time

Success Scenario

	Actor Action		System Response
1	Click on Node Management → Add Node menu	2	Add node dialog box opens
3	User enters the input	4	Input is validated.
5	User pressed the 'Add' button	6	Node is added

Failure Scenario

	3a Display message 'Invalid Input'

e. Delete Node

Name:	Delete Node use case
Actor:	Cluster User
Type:	Primary, Concrete, Essential
Overview:	The user can delete the cluster nodes by selecting them from the list.
Triggering Event:	User invokes "Delete Nodes" from the 'Node Management' menu
Precondition:	Cluster is initialized and a node is selected
Post Condition	The selected node is deleted and is not a part of the cluster anymore
Constraints:	The user may select one node at a time

Success Scenario

	Actor Action		System Response
1	Click on Node Management → Delete Node menu	2	Delete Node dialog box opens
3	User makes a Selection and presses the Delete button	4	The requested option is processed and the specified node is deleted.

Failure Scenario

	4a User did not select any node

f. Memory Monitor

Name:	Memory Monitor use case
Actor:	Cluster User
Type:	Primary, Concrete, Essential
Overview:	The user Monitors the Memory
Triggering Event:	User invokes "Memory Monitor" from the 'Node Management' menu
Precondition:	The cluster is initialized
Post Condition:	The 'Memory Monitor' runs.
Constraints:	None

Success Scenario

	Actor Action		System Response
1	Click on Node Management → Memory Monitor menu	2	Memory Monitor dialog box opens

No Failure Scenario

g. Run IDE

Name:	Run IDE use case
Actor:	Cluster User
Type:	Primary, Concrete, Essential
Overview:	This use case opens the Cluster IDE
Triggering Event:	Select the Run IDE menu
Precondition:	The Cluster is running
Post Condition	The IDE application is opened
Constraints:	None

Success Scenario

	Actor Action		System Response
1	Click Run Application→ Run IDE menu	2	Opens the IDE dialog box
3	Click on the button 'MPICH IDE'	4	Opens MPICH IDE

No Failure Scenario

h. IDE

Name:	IDE use case
Actor:	Cluster User
Type:	Secondary
Overview:	This use case invokes the IDE
Triggering Event:	User chooses 'MPICH IDE' from 'Run IDE'
Precondition:	Cluster Management System Software is running
Post Condition	IDE Software Opens
Constraints:	None

Success Scenario

	Actor Action		System Response
1	Select Run IDE menu → MPICH IDE	2	IDE is invoked.

No Failure Scenario

3.8.2 Use Cases of Integrated Development Environment

The use cases for the Integrated Development Environment are as follows:

a. File New

Name:	File New use case
Actor:	Cluster User
Type:	Primary, Concrete, Essential
Overview:	The user wishes to open a new file named as 'untitled.c' to work on. The file is opened on his request.
Triggering Event:	Either the menu item is selected or the button is pressed
Pre Condition:	IDE is running
Post Condition	A new file is opened
Constraints	None

Success Scenario

	Actor Action		System Response
1	Select File → New or press the new button	2	The file opens

No Failure Scenario

b. File Open

Name:	File Open use case
Actor:	Cluster User
Type:	Primary, Concrete, Essential

Overview:	The user can open the file he wanted by invoking this use case
Triggering Event:	Either the menu item is selected or the button is pressed
Pre Condition:	The IDE is running
Post Condition	The specific file opens
Constraints	None

Success Scenario

	Actor Action		System Response
1	Select File→ Open or press the open button on the tool bar	2	The specific file is opened

No Failure Scenario

c. File Save

Name:	File Save use case
Actor:	Cluster User
Type:	Primary, Concrete, Essential
Overview:	After the file is opened, it can be saved using this use case
Triggering Event:	Either the menu item is selected or the button is pressed
Pre Condition:	File is Opened
Post Condition	File is Saved
Constraints	None

Success Scenario

	Actor Action		System Response
1	Select File → Save menu item or press Save button on the tool bar.	2	Sees if a file is opened to be saved
		3	Sees if the File is untitled or is already saved somewhere.
		4	If untitled, 'Save as' dialog box is opened
		5	If already saved, Saves under the same name.

Failure Scenario

	2a No file is Opened

d. File Save As

Name:	Save As use case
Actor:	Cluster user
Type:	Primary, Concrete, Essential
Overview:	Saves the file under the name specified by the user.
Triggering Event:	Either the menu item is selected or the button is pressed

Precondition:	A file is opened
Post Condition	The file is Saved under the name specified by the user

Success Scenario

	Actor Action		System Response
1	Select File → Save As menu or press the button on the tool bar	2	Opens the 'Save As' dialog box
		3	Takes the input of the file name
		4	Saves the File

Failure Scenario

	2a No File is opened to be saved
	3a Displays Message 'Please specify the name of the File'
	3b if the file already exists, asks the user if it should replace the file. If not, the user is asked to specify another name

e. File Edit

Name:	File Edit use case
Actor:	Cluster User
Type:	Primary, Concrete, Essential
Overview:	Enables the user to use the properties of editing
Triggering Event:	Either the menu item is selected or the button is pressed
Pre Condition:	File is open
Post Condition	The selected text is edited
Constraints	None

Success Scenario

	Actor Action		System Response
1	Select Edit → Cut/Copy/Paste	2	Cuts/Copies/Pastes the selected text

Failure Scenario

	2a User did not select any text to be edited

f. Select Text

Name:	Edit Select all use case
Actor:	Cluster User
Type:	Primary, Concrete, Essential
Overview:	Enables the user to use the property 'Select all' of editing
Triggering Event:	The menu item is selected
Pre Condition:	There is some text present in the text area
Post Condition	All the available text is selected
Constraints	None

Success Scenario

	Actor Action		System Response
1	Select Edit → Select All	2	Selects all the text present

Failure Scenario

	2a There is no text to be selected

g. Compile

Name:	Compile use case
Actor:	Cluster User
Type:	Primary, Concrete, Essential
Overview:	This case describes how a program is compiled in the IDE.
Triggering Event:	Either the menu item is selected or the button is pressed
Pre Condition:	File is open
Post Condition	File is compiled
Constraints	None

Success Scenario

	Actor Action		System Response
1	Select Run → Compile or press the 'Compile' button	2	Compiles the File
		3	Displays the 'Compiled successfully' message in the tool bar.

Failure Scenario

	2a Compilation Error

h. Run

Name:	Run use Case
Actor:	Cluster User
Overview:	This use case runs the compiled file
Type:	Primary, Abstract, Essential
Triggering Event:	Either press the button or choose from the menu
Pre Condition:	File should be compiled
Post Condition	Output displayed
Constraints:	None

Success Scenario

	Actor Action		System Response
1	Click the Run → Run Menu or press the "Run" button on toolbar	2	Runs the compiled program.

Failure Scenario

	2a Runtime/Fatal Errors.

i. Set Properties

Name:	Properties use case
Actor:	Cluster User
Overview:	This use case is used to specify the number of nodes the Cluster aware Application should run on
Type:	Primary, Abstract, Essential
Triggering Event:	Choose from the menu
Pre Condition:	-
Post Condition	Number of nodes specified
Constraints:	None

Success Scenario

	Actor Action		System Response
1	Choose Run → Properties	2	Opens the Properties dialog box
		3	Validates the input
		4	Changes the number of nodes on user's request

Failure Scenario

	3a Message is displayed 'Invalid Input'

CHAPTER 4

SYSTEM DESIGN

4 System Design

System design is the specification or construction of a technical, computer-based solution for the business requirements identified in the system analysis. It is the evaluation of alternative solutions and the specification of a detailed computer-based solution. The design phase is the first step towards moving from problem domain to the solution domain. System design develops the architectural detail required to build a system or product.

4.1 Object-Oriented Design Method

Object-Oriented design translates the OOA model of the real world into an implementation-specific model that can be realized in software. Object-oriented design transforms the analysis model, created using object-oriented analysis method, into a design model that serves as a blueprint for software construction. For the development of the system under consideration the same technique is used.

Object-oriented design (OOD) is concerned with developing an object-oriented model of a software system to implement the identified requirements. Many OOD methods have been described since the late 1980s. The most popular OOD methods include Booch, Buhr, Wasserman, and the HOOD method developed by the European Space Agency. Object-oriented design (OOD) is concerned with developing an object-oriented model of a software system to implement the identified requirements. Many OOD methods have been described since the late 1980s. The most popular OOD methods include Booch, Buhr, Wasserman, and the HOOD method developed by the European Space Agency.

OOD can yield the following benefits: *maintainability* through simplified mapping to the problem domain, which provides for less analysis effort, less complexity in system design, and easier verification by the user; *reusability* of the design artifacts, which saves time and costs; and productivity gains through direct mapping to features of Object-Oriented Programming Languages.

OOD builds on the products developed during Object-Oriented Analysis (OOA) by refining candidate objects into classes, defining message protocols for all objects, defining data structures and procedures, and mapping these into an object-oriented programming language (OOPL).

4.2 UML, A Unified Approach To Object Oriented Design

Design can be thought of in two phases. The first, called high-level design, deals with the decomposition of the system into large, complex objects. The second phase is called low-level design. In this phase, attributes and methods are specified at the level of individual objects. This is also where a project can realize most of the reuse of object-oriented products, since it is possible to guide the design so that lower-level objects correspond exactly to those in existing object libraries or to develop objects with reuse potential. As in OOA, the OOD artifacts are represented using CASE tools with object-oriented terminology.

Like object-oriented analysis, there are many different object-oriented design methods. UML is an attempt to provide a single approach to OOD that is applicable in all application domains. UML approaches the design process through two levels of abstraction; design of subsystems (architecture) and design of individual objects.

4.3 Activity Diagrams

The activity diagrams of the project are as follows:

4.3.1 Activity Diagrams of Cluster Management System

After a thorough study of Cluster Management System, the following activity diagrams were formed for the CMS:

- Initialize Cluster
- Add Node
- Delete Node

a. Activity Diagram of Initialize Cluster

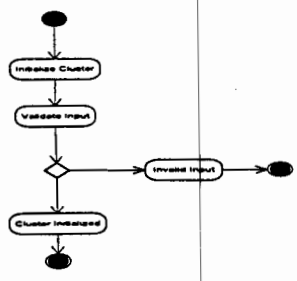


Fig4.01: Activity Diagram of Initialize Cluster

c. Activity Diagram of Delete Node

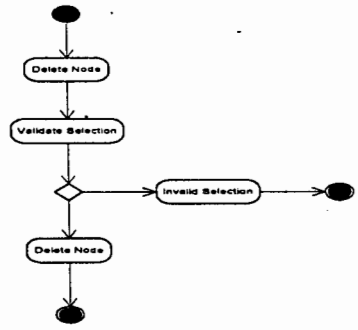


Fig4.03: Activity Diagram of Delete Node

4.3.2 Activity Diagrams of Integrated Development Environment

The activity diagrams for the IDE are as follows:

- File New/Open
- File Save
- File Edit
- Compile File
- Run File
- Set Properties

c. Activity Diagram of Delete Node

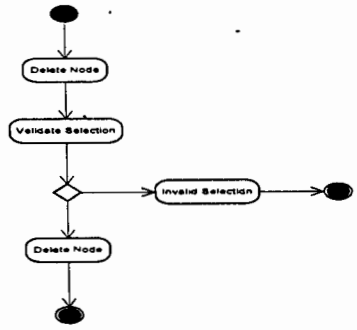


Fig4.03: Activity Diagram of Delete Node

4.3.2 Activity Diagrams of Integrated Development Environment

The activity diagrams for the IDE are as follows:

- File New/Open
- File Save
- File Edit
- Compile File
- Run File
- Set Properties

a. Activity Diagram of File New/Open

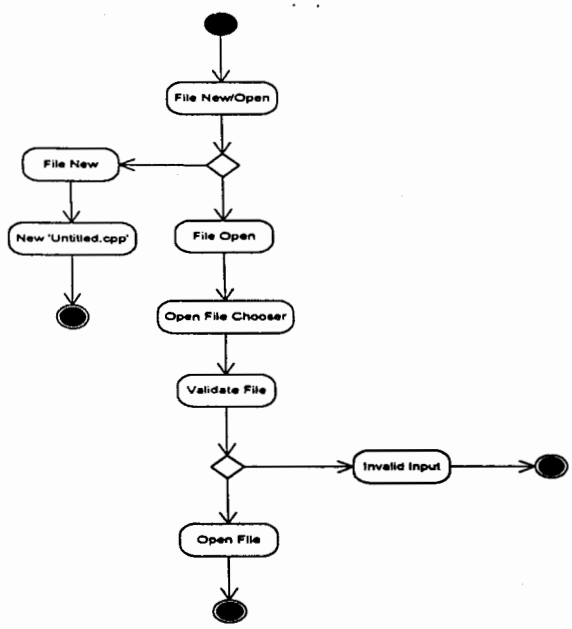


Fig4.04: Activity Diagram of File New/Open

b. Activity Diagram of File Save

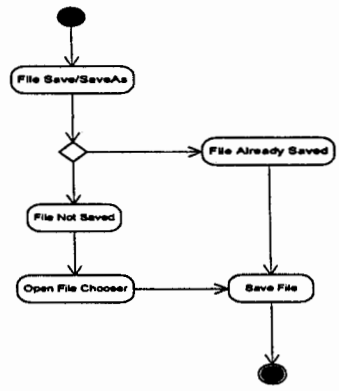


Fig4.05: Activity Diagram of File Save

c. Activity Diagram of File Edit

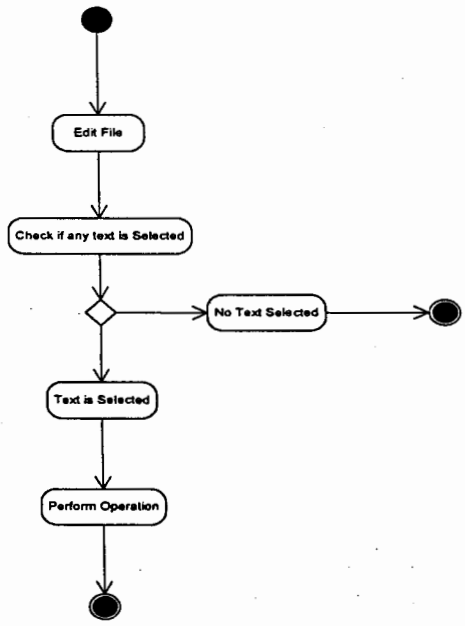


Fig4.06: Activity Diagram of File Edit

d. Activity Diagram of Compile File

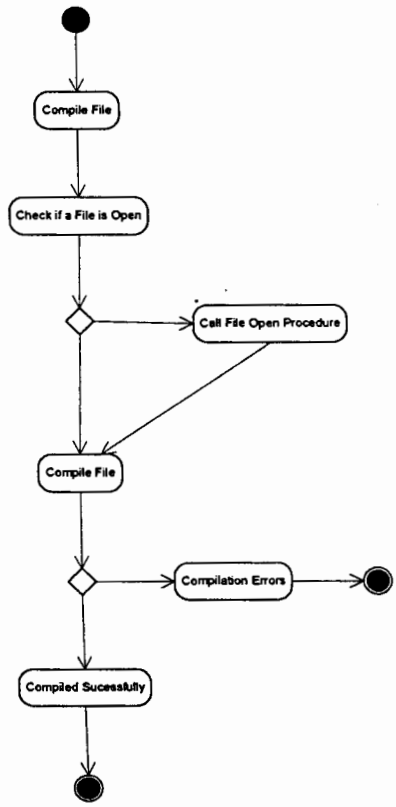


Fig4.07: Activity Diagram of Compile File

e. Activity Diagram of Run File

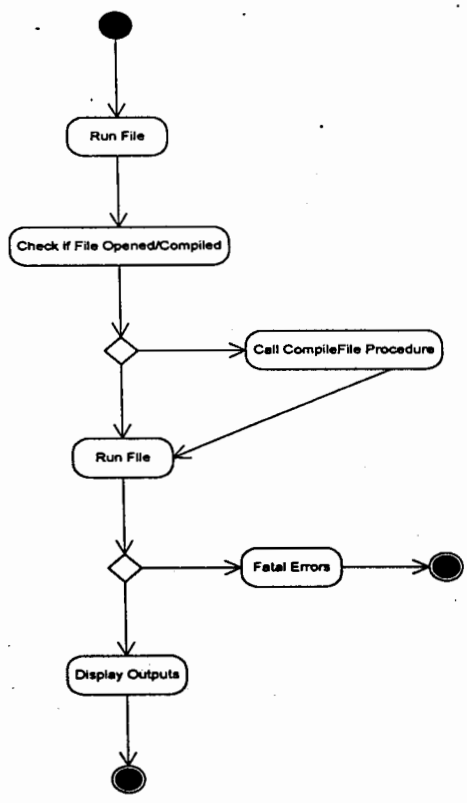


Fig4.08: Activity Diagram of Run File

f. Activity Diagram of Set Properties

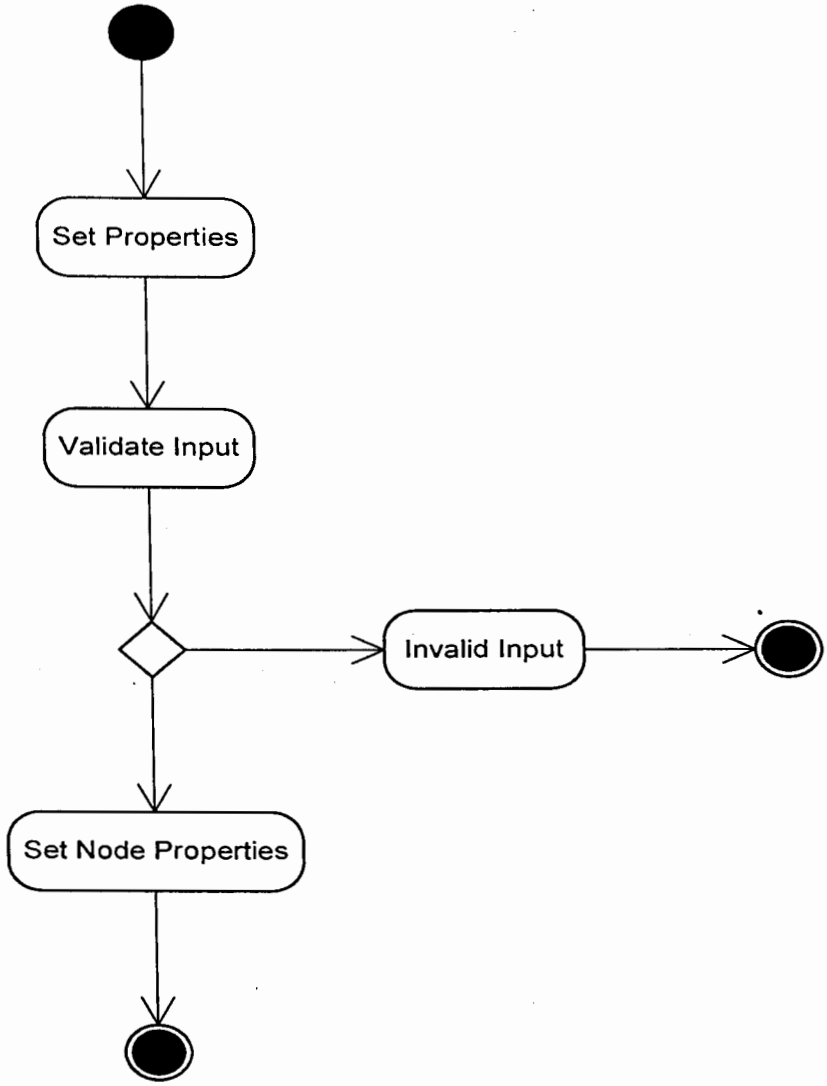


Fig4.09: Activity Diagram of Set Properties

4.4 Sequence Diagrams

Sequence diagrams document the interactions between classes to achieve a result, such as a use case. Because UML is designed for object-oriented programming, these communications between classes are known as messages. The Sequence diagram lists objects horizontally, and time vertically, and models these messages over time. Sequence Diagrams are about deciding and modeling how a system will achieve what was described in the Use Case model. The advantage of developing sequence diagrams is that they help in the forming class models, and thus leading to the actual implementation. A sequence diagram has the objects, messages, actors, lifelines, and activations; classes and actors are listed as columns, with vertical lifelines indicating the lifetime of the object over time.

Objects are instances of classes, and are arranged horizontally. The pictorial representation for an object is a class (a rectangle) with the name prefixed by the object name (optional) and a semi-colon.

Actors can also communicate with objects, so they too can be listed as a column. An Actor is modeled using the ubiquitous symbol, the stick figure.

The Lifeline identifies the existence of the object over time. The notation for a Lifeline is a vertical dotted line extending from an object.

Activations, modeled as rectangular boxes on the lifeline, indicate when the object is performing an action.

Messages, modeled as horizontal arrows between activations, indicate the communication between objects.

4.4.1 Sequence Diagrams of Cluster Management System

After a thorough study of Cluster Management System, the following sequence diagrams were formed:

- Initialize Cluster
- Shutdown Cluster
- Restart Cluster
- Add Node
- Delete Node
- Run IDE

a. Sequence Diagram of Initialize Cluster

The following diagram is true for subsequent figures as well.

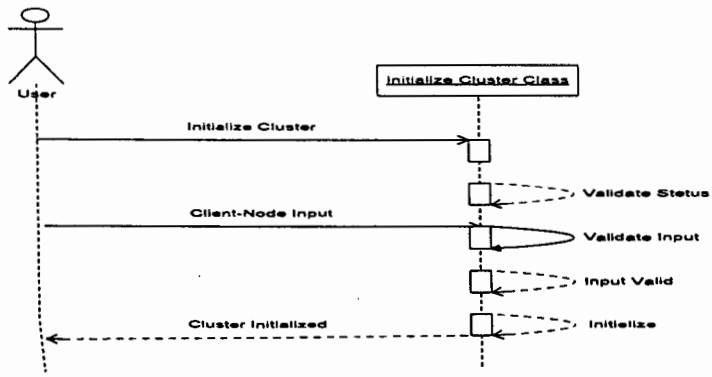


Fig 4.10: Sequence Diagram of Initialize Cluster

b. Sequence Diagram of Shutdown Cluster

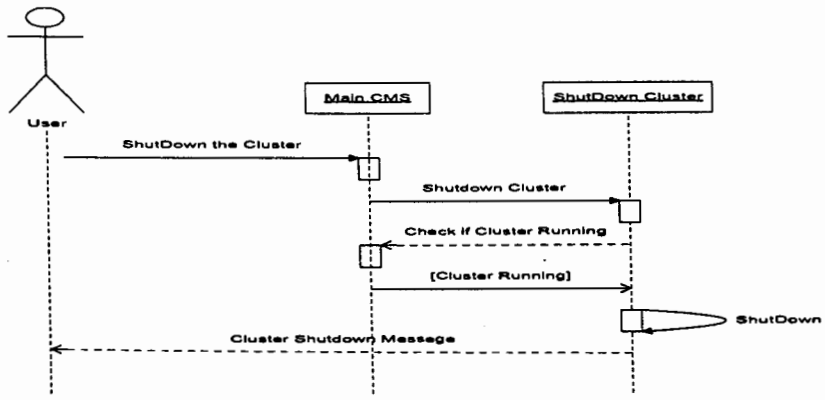


Fig4.11: Sequence Diagram of Shutdown Cluster

c. Sequence Diagram of Restart Cluster

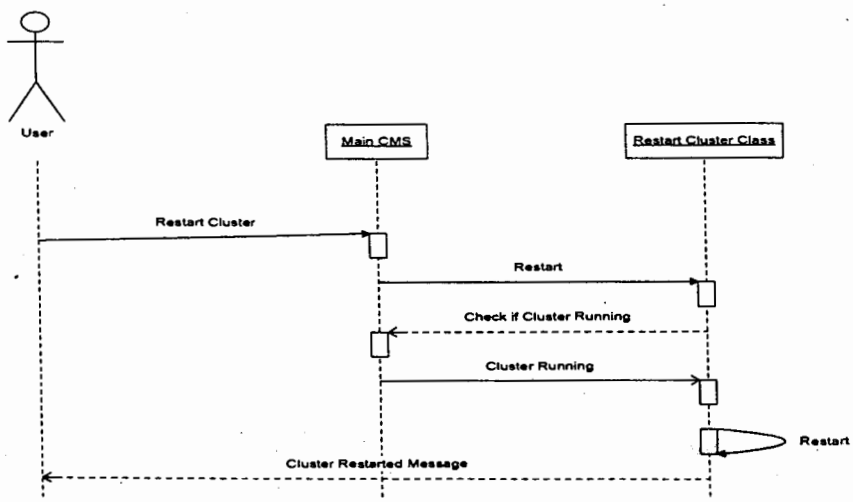


Fig 4.12: Sequence Diagram of Restart Cluster

d. Sequence Diagram of Add Node

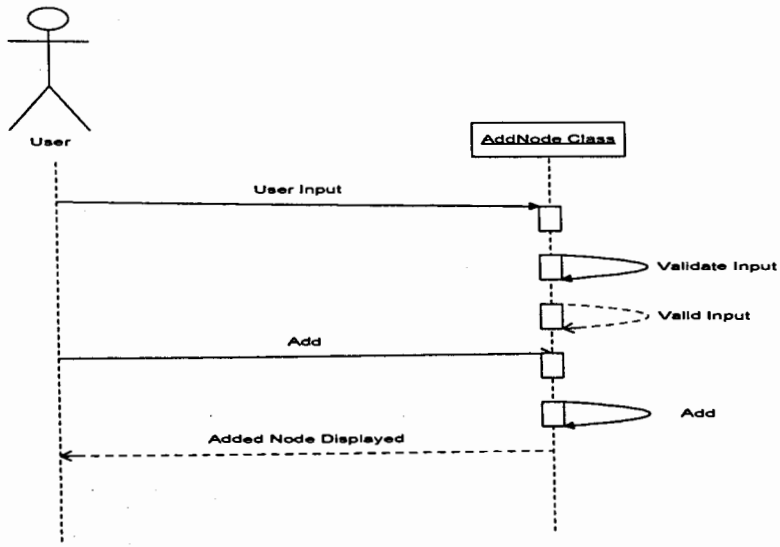


Fig 4.13: Sequence Diagram of Add Node

e. Sequence Diagram of Delete Node

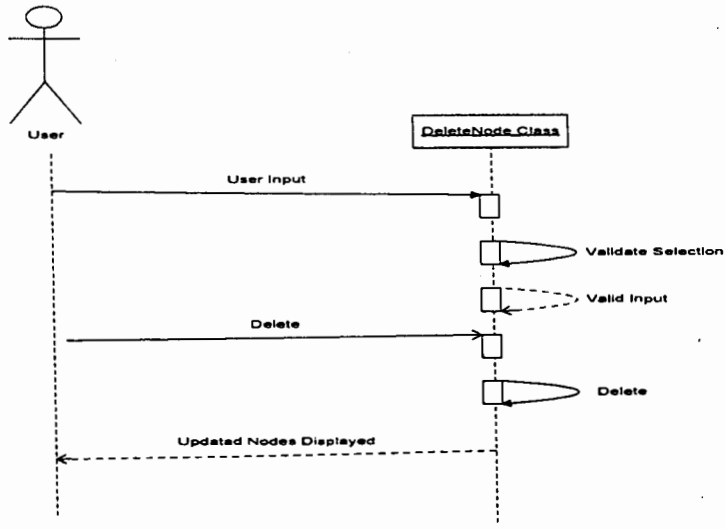


Fig 4.14: Sequence Diagram of Delete Node

f. Sequence Diagram of Run IDE

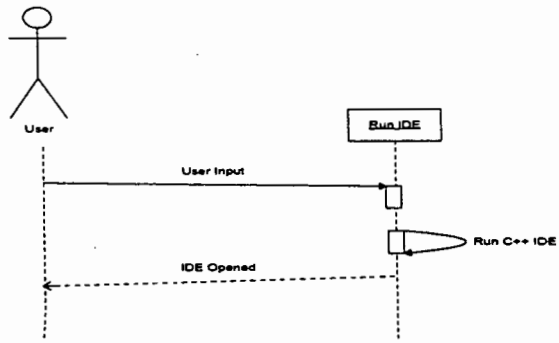


Fig 4.15: Sequence Diagram of Run IDE

4.4.2 Sequence Diagrams of Integrated Development Environment

The sequence diagrams of the IDE are as follows:

- File New
- File Open
- File Save
- File Edit
- Compile File
- Run File
- Set Properties

a. Sequence Diagram of File New

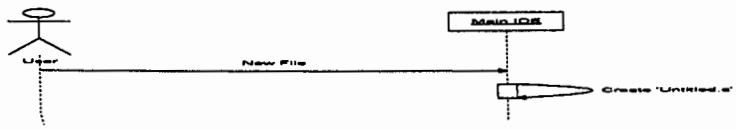


Fig4.16: Sequence Diagram of File New

b. Sequence Diagram of File Open

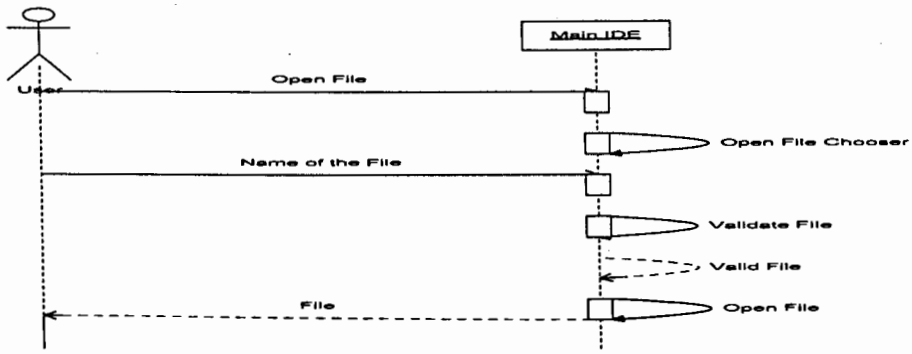


Fig4.17: Sequence Diagram of File Open

c. Sequence Diagram of File Save

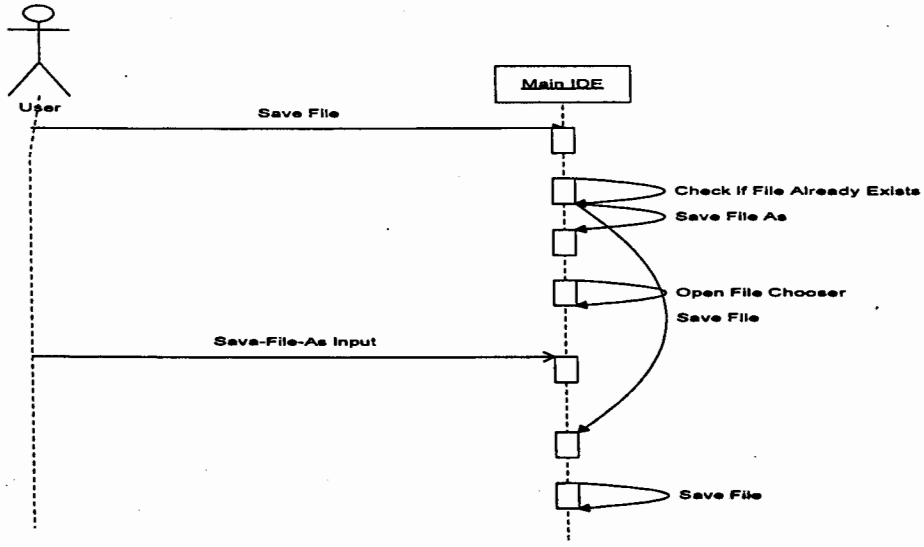


Fig4.18: Sequence Diagram of File Save

d. Sequence Diagram of File Edit

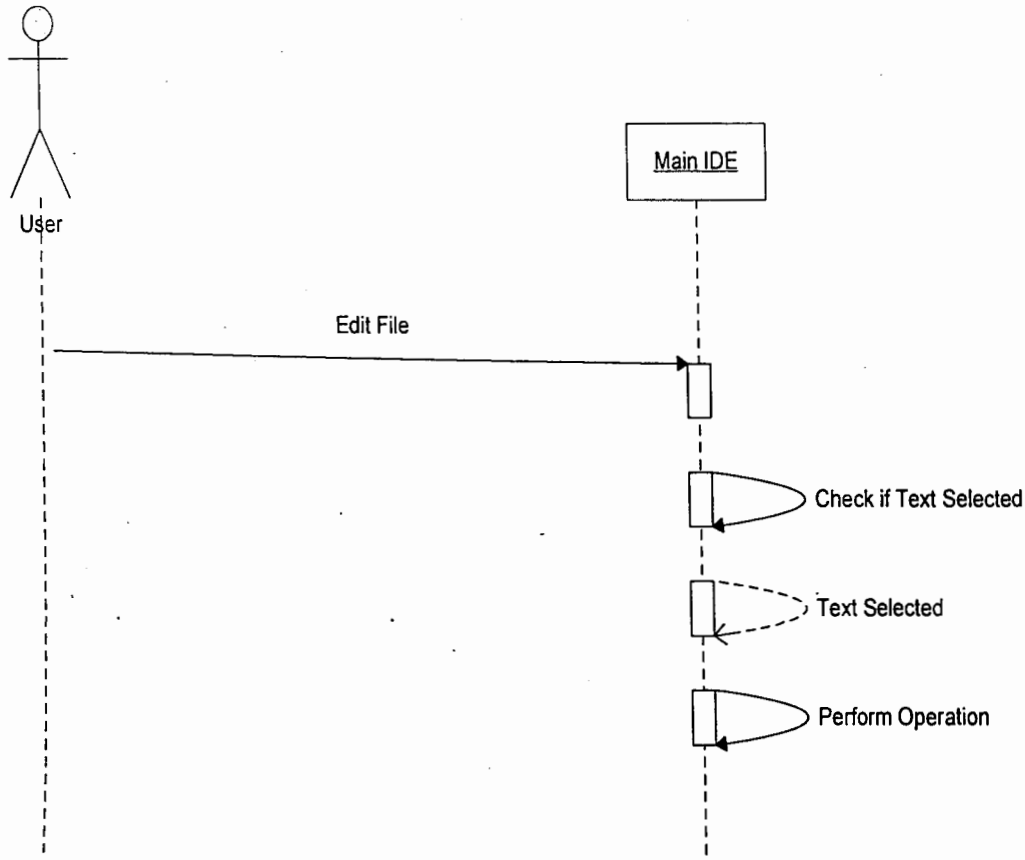


Fig4.19: Sequence Diagram of File Edit

e. Sequence Diagram of Compile File

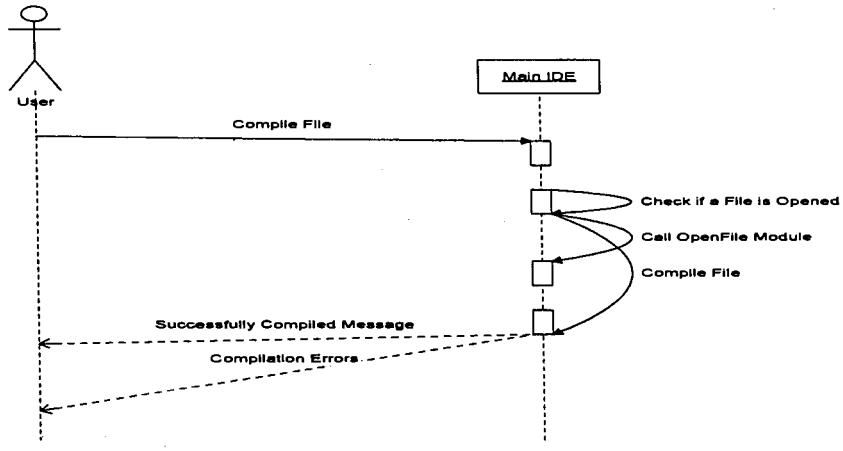


Fig4.20: Sequence Diagram of Compile File

f. Sequence Diagram of Run File

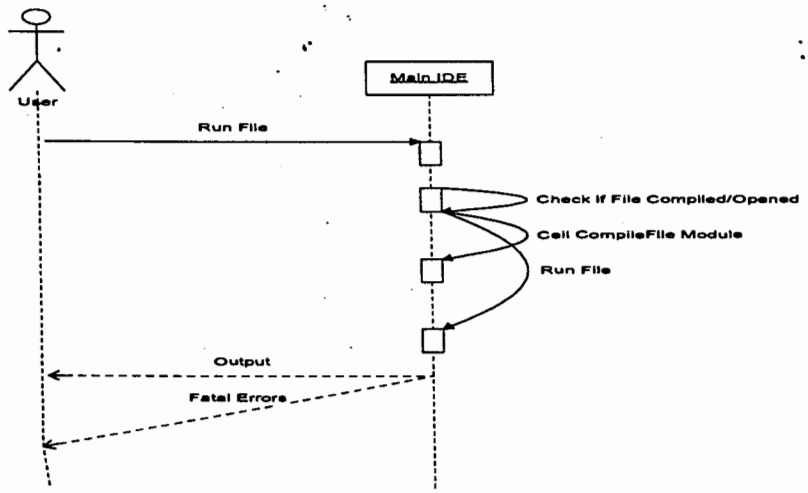


Fig4.21: Sequence Diagram of Run

g. Sequence Diagram of Set Properties

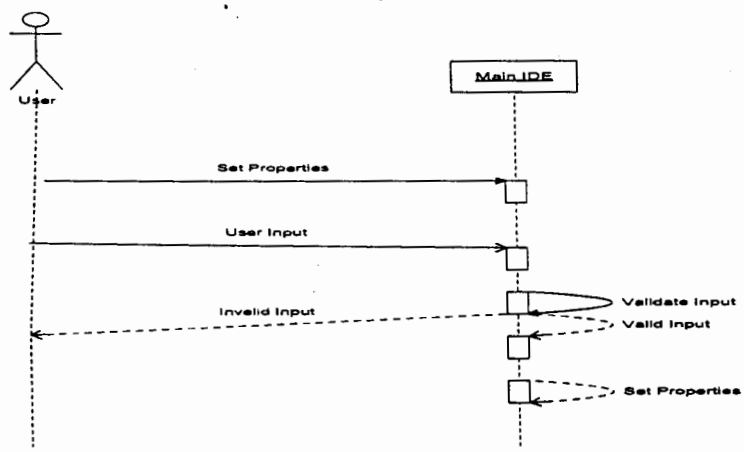


Fig4.22: Sequence Diagram of Set properties

4.5 Class Diagrams

Class diagrams are the backbone of almost every object-oriented method including UML. It can be said that class diagrams identify the class structure of a system, including the properties and methods of each class. Also depicted are the various relationships that can exist between classes, such as an inheritance relationship

Another purpose of class diagrams is to specify the class relationships and the attributes and behaviors associated with each class. Class diagrams are remarkable at illustrating inheritance and composite relationships. A class diagram consists of one major component and that is the various classes, along with these are the various relationships shown between the classes such as aggregation, association, composition, dependency, and generalization.

Classes are the building blocks in object-oriented programming. A Class is depicted using a rectangle divided into three sections. The top section is the name of the Class. The middle section defines the properties of the Class. The bottom section lists the methods of the class.

An Association is a generic relationship between two classes, and is modeled by a line connecting the two classes. This line can be qualified with the type of relationship, and can also feature multiplicity rules (e.g. one-to-one, one-to-many, many-to-many) for the relationship.

If a class cannot exist by itself, and instead must be a member of another class, then that class has a Composition relationship with the containing class. A line with a filled diamond indicates a Composition relationship.

When a class uses another class, perhaps as a member variable or a parameter, and so "depends" on that class, a Dependency relationship is formed. A dotted arrow indicates a Dependency relationship.

Aggregations indicate a whole-part relationship, and are known as "has-a" relationships. A line with a hollow diamond indicates an Aggregation relationship.

A Generalization relationship is the equivalent of an *inheritance* relationship in object-oriented terms (an "is-a" relationship). An arrow indicates a Generalization relationship with a hollow arrowhead pointing to the base, or "parent", class.

4.5.1 Class Diagram of the Project

Following are the classes identified in this project:

- ClusterManagementSystem
- InitializeClusterDialog
- ShutDownDialog
- RestartClusterDialog
- AddNodeDialog

- DeleteNodeDialog
- RunIDEDialog
- MemoryMonitorPanel
- Serveride
- SaveFileFilter
- PropDialog

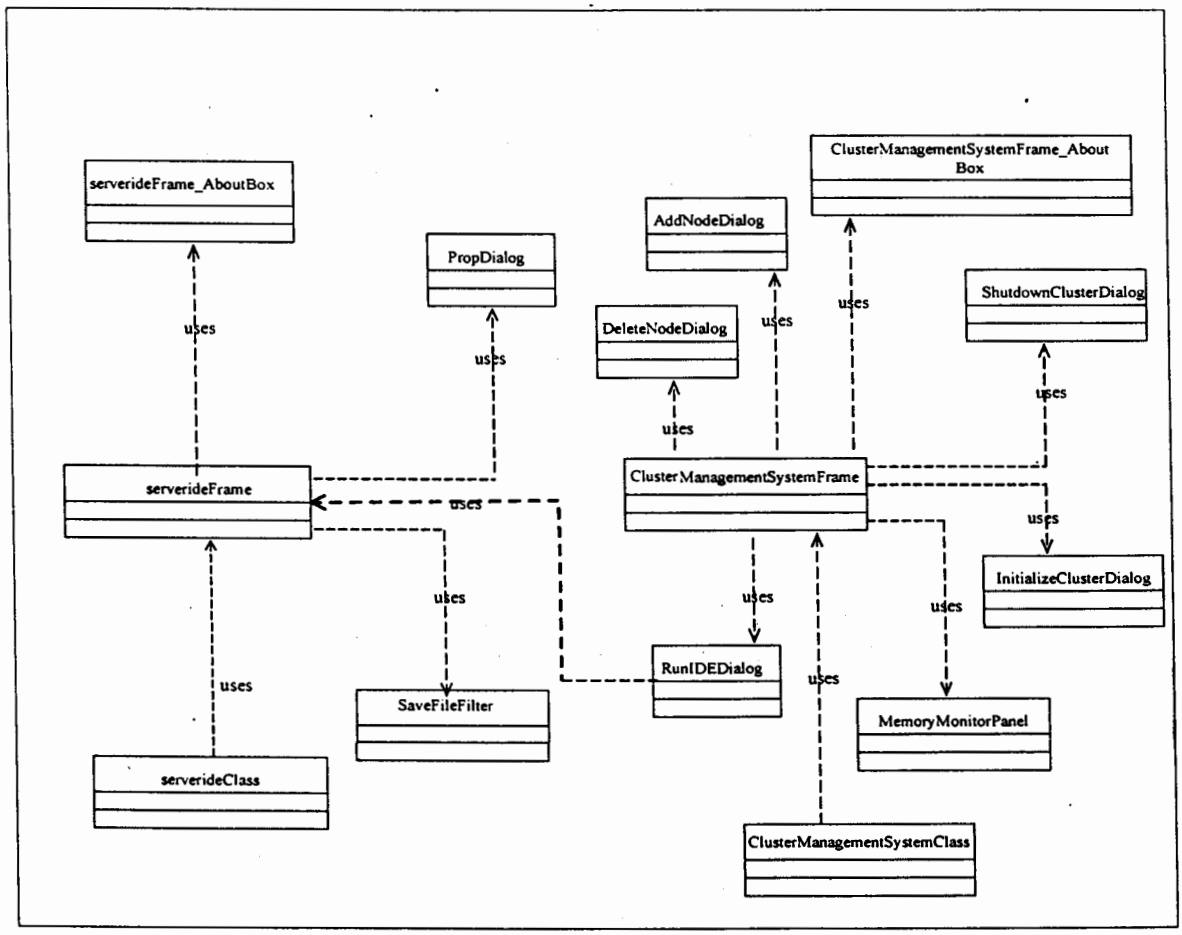


Fig 4.23: Class Diagram of Cluster Management System

CHAPTER 5

IMPLEMENTATION

5 Tools and Technologies

This chapter contains the description of the tools and technologies used for the project. This helps the user for better understanding of the software.

5.1 Languages Employed for Development

The languages used for the development of Cluster Management System targeted for Linux Red Hat 8.0 are:

5.1.1 Java 2 Standard Edition (J2SE)

The Java platform consists of the Java application programming interfaces (APIs) and the Java virtual machine (JVM).

Java APIs are libraries of compiled code that you can use in your programs. They let you add ready-made and customizable functionality to save you programming time.
The simple program in this lesson uses a Java API to print a line of text to the console. The console printing capability is provided in the API ready for you to use; you supply the text to be printed.

Java programs are run (or interpreted) by another program called the Java VM. If you are familiar with Visual Basic or another interpreted language, this concept is probably familiar to you. Rather than running directly on the native operating system, the program is interpreted by the Java VM for the native operating system. This means that any computer system with the Java VM installed can run Java programs regardless of the computer system on which the applications were originally developed.

Java has had trouble gaining acceptance in a desktop PC environment due to current speed limitations, as applications are run through a virtual machine. But the language has proven itself to be more beneficial to the wireless platform, taking advantage of Java's code reuse and small display capabilities.

Hand held, battery-operated products such as cell phones, two-way pagers and personal organizers communicate with other devices by way of a wireless link. The Java programming language empowers developers to write device-independent wireless applications for accessing, downloading, or updating information over the wireless link.

A highly optimized Java runtime environment, J2SE technology specifically addresses the vast consumer space, which covers the range of extremely tiny commodities such as smart cards or a pager all the way up to the set-top box, an appliance almost as powerful as a computer.

J2SE is short for Java 2, Standard Edition and it is the Java platform specifically created for small, constrained devices - devices such as cell phones, PDAs and embedded dedicated devices such as set-top or automotive computers.

5.2 Tools Used

Our project, being an attempt to introduce the evolving field of parallel computing, we have developed the Cluster Management System and Integrated Development Environment using the following tools:

5.2.1 JBuilder 8 Enterprise Edition

Borland JBuilder 8 Enterprise Edition's built-in productivity tools, now featuring support for UML diagrams and unit testing, make it a good investment for Java development shops. This Java IDE is a good investment for enterprises that want to increase developer productivity and reduce project time lines and costs. Newly added support for modeling diagrams, integrated unit test capabilities, and built-in tools for code documentation will help streamline Java projects in the enterprise.

Borland has done a good job of integrating multiple tools into the latest version of its Java IDE (integrated development environment), JBuilder 8 Enterprise. With this release, Java developers can write code, debug it, and deploy it, as usual. They also can use built-in UML (Unified Modeling Language) tools to strengthen project designs, leverage integrated tools for building and running unit tests, and take advantage of included wizards that help generate Java documentation (Javadoc files) from code.

Borland has integrated the JUnit unit testing framework within the JBuilder 8 IDE and has included a number of useful test facilities that should help reduce test creation and execution times.

JBuilder supports rename and move refactoring. Rename refactoring allows developers to change the name of a package, class, field, local variable, or method and ensure that any references to the newly named item are updated correctly. Move refactoring, available only for classes, allows coders to move a top-level public class to a new package.

Altogether, JBuilder 8 is a good investment for enterprises that want to boost programmer productivity and decrease development costs. Whether you are upgrading from a previous release or purchasing it for the first time, JBuilder 8 is well worth deploying in the enterprise.

5.2.2 MPICH 1.2.5.2:

This message-passing model of parallel computation has emerged as an expressive, efficient, and well-understood paradigm for parallel programming. Until recently, the syntax and precise semantics of each message-passing library implementation were different from the others, although many of the general semantics were similar. The proliferation of message-passing library designs from both vendors and users was appropriate for a while, but eventually it was seen that enough consensus on requirements and general semantics for message-passing had been reached that an attempt at standardization might usefully be undertaken.

The process of creating a standard to enable portability of message-passing applications codes began at a workshop on Message Passing Standardization in April1992, and the Message Passing Interface (MPI) Forum organized itself at the Super coming '92 Conference. During the next eighteen months Version 1.0 of the MPI Standard was completed. Important contributions came from Zip code, Chimp, PVM, Chameleon, and PICL.

5.2.2.1 Features of MPI

MPI included point-to-point message passing and collective operations, all scoped to a user-specified group of processes. Communications, which house groups and communications context information, provide an important measure of safety that is necessary and useful for building up library-oriented parallel code.

MPI also provides three additional classes of services: environmental inquiry, basic timing information monitoring. MPI makes heterogeneous data conversion a transparent part of its services by requiring data-type specifications for all communication operations. Both built-in and user-defined data-types are provided.

MPI accomplishes its functionality with opaque objects, with well-defined constructors and destructors, giving MPI an object-based look and feel. Opaque objects include groups, communicators, and request objects for asynchronous operations. User-defined and predefined data types allow for heterogeneous communication and elegant description of gather/scatter semantics in send/receive operations as well as collective operations.

MPI provides support for both the SPMD and MPMD modes of parallel programming. Furthermore, MPI can support inter-application computations through inter-communicator operations, which support communication between groups rather than within a single group. Dataflow-style computations also can be constructed from inter-communicators. MPI provides a thread-safe application-programming interface (API), which will be useful in multithreaded environments as implementations mature and support thread safety themselves.

5.2.2.2 Implementation of MPI

The project to provide a portable implementation of MPI began at the same time as the MPI definition process itself. The idea was to provide early feedback on decisions being made by the MPI Forum and provide an early implementation to allow users to experiment with the definitions even as they were being developed. Targets for the implementation were to include all systems capable of supporting the message-passing model. MPICH is a freely available, complete implementation of the MPI specification, designed to be both portable and efficient. The "CH" in MPICH stands for "Chameleon", symbol of adaptability to one's environment and thus of portability. Chameleon are fast, and from the beginning a secondary goal was to give up as little efficiency as possible for the portability.

MPICH is both a research project and a software development project. As s research project, its goal is to explore methods for narrowing the gap between the programmer of a parallel

computer and the performance deliverable by its hardware. In MPICH, we adopt the constraint that the programming interface will be MPI, reject constraints on the architecture of the target machine, and retain high performance (measured in terms of bandwidth and latency for message-passing operations) as a goal. As a software project, MPICH's goal is to promote the adoptation of the MPI Standard by providing users with a free, high-performance implementation on a diversity of platforms, while aiding vendors in providing their own customized implementations.

5.2.2.3 Systems Supported by MPICH

MPICH supports a wide range of systems. These include:

i) Workstation clusters, running various versions of Unix, including but not limited to AIX Digital Unix, FreeBSD, HP-UX, IRIX, LINUX, Solaris, and SunOS.
ii) Windows NT and Windows 2000.
iii) IBM SP
iv) Intel i860, Delta and Paragon.
v) Shared Memory systems (SMP) including HP/Convex Exemplar
vi) CRAY T3D

5.2.2.4 Architecture of MPICH

The software architecture of MPICH supports the conflicting goals of portability and high performance. The design is guided by two principles. First, to maximize the amount of code that can be shared without compromising performances. A large amount of the code in any implementation is system independent. Implementation of most of the MPI opaque objects, including data-types, groups, attributes, and even communicators, is platform-independent. Many of the complex communication operations can be expressed portably in terms of lower-level ones. Second principal is to provide a structure whereby MPICH could be ported to a new platform quickly, and then gradually tuned for that platform by replacing parts of the shared code by platform-specific code.

The central mechanism for achieving the goals of portability and performance is a specification called the Abstract Device Interface (ADI). All MPI functions are implemented in terms of the macros and functions that make up the ADI. All such code is portable. Hence, MPICH contains many implementations of the ADI, which provide portability, ease of implementation, and an incremental approach to trading portability for performance. One implementation of the API is in terms of a lower level interface called the channel interface. The channel interface can be extremely small and provides the quickest way to port MPICH to a new environment. Such a port can then be expanded gradually to include specialized implementation of more of the API functionality. The architecture decisions in MPICH are those that relegate the implementation of various functions to the channel interface, the ADI, or the application programmer interface (API), which in this case is MPI.

5.2.3 Dynamics Application

The problem of unsteady-state flow of heat is a physical situation that can be represented by a parabolic partial-differential equation. The simplest situation is for flow of heat in one direction. Imagine a rod that is uniform in cross section and insulated around its perimeter so that heat flows only longitudinally. Consider a differential portion of the rod, *dx* in length with cross-sectional area A. We let *u* represent the temperature at any point in the rod, whose distance from left end is x. heat is flowing from left to right under the influence of the temperature gradient du/dx. The difference between the rate of flow in and rate of flow out is the rate at which heat is being stored in the element. If *c* is the heat capacity, and *p* is the density, *t* for time we have a simplified form

$$k\,(d^2u/dx^2) = c\rho\,(du/dt)$$

This is the basic mathematical model for unsteady-state flow. It has been derived from heat flow. [1] But it applies equally to diffusion of material, flow of fluids (under conditions of laminar flow), flow of electricity in cables (the telegraph equation), and so on. The function that we call the solution to the problem not only must obey the differential equation given above , but also must satisfy an initial condition and a set of boundary conditions. For one-dimensional heat flow problem we first consider, the initial condition will be the initial temperature at all points along the rod,

$$U(x,t) = u(x,0) = f(x). \quad \text{at } t=0$$

the boundary conditions will describe the temperature at each end of the rod as functions of time. We make these temperatures constant:

$$u(0,t) = c1$$
$$u(L,t) = c2$$

now we see the application of finite differences to solve parabolic partial-differential equations.

The Explicit Method

Divides space and time into discrete uniform subintervals and replaces both time and space derivatives by finite-difference approximations, permitting one to easily compute values of the function at a time *dt* after the initial time. These values are then used to compute a second set of values and the process is repeated. One approach to solving parabolic partial-differential equations by a numerical method is to replace the partial derivatives by finite-difference approximations. For the one-dimensional heat-flow equation,

$$d^2u/dx^2 = (c\rho/k)(du/dt)$$

eq(5.1)

We can use the relation

$$d^2u/dx^2 = u_{i+1}^{j} - 2u_i^j + u_{i-1}^{j}/(dx)^2 + O(dx)^2$$

eq(5.2)

$$du/dt = ui\ ^{\wedge}\ (j+1) - u\ i\ ^{\wedge}\ j\ /\ dt\ +\ O\ (dt)$$

eq(5.3)

Here subscript is used to denote position and superscripts for time.
Finally substituting eq (5.2) and (5.3) into (5.1) and solving for *u i ^ (j+1)* gives the equation for the forward-difference method:

$$U\ i\ ^{\wedge}\ (j+1) = r\ (U\ i+1\ ^{\wedge}\ j + U\ i-1\ ^{\wedge}\ j) + (1 - 2\ r)\ U\ i\ ^{\wedge}\ j$$

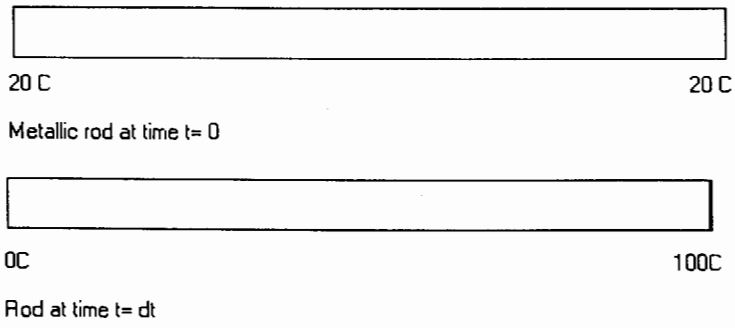
eq(5.4)

where r is

$$r = k\ \partial t\ /\ c\ p\ (\partial x)\ ^{\wedge}\ 2$$

5.2.3.1 Application

We have solved for *U I (j+1)* in terms of the temperatures at time *tj* in eq (5.4) in view of the normally known conditions for a parabolic partial-differential equation. We subdivide the length into uniform subintervals and apply our finite-difference approximation to Eq (5.1) at each point where *u* is not known. Equation (5.4) then gives the values of *u* at each interior point at *t=t1* since the values at $t = t^{\circ}$ are given by the initial conditions. It can then be used to get values at t_2 using the values at t_1 as initial conditions, so we can step the solution forward in time. At the endpoints, the boundary conditions will determine *u*. The relative size of the time and distance steps, *dt* and *dx*, affects Eq (5.4). The efficiency of related computer program depends upon the size of time steps; reducing the value of *dt* requires more successive calculations to reach a given time after the start of heat flow. We can save our resources by executing our program on other available nodes of the cluster. Moreover the execution time would be reduced if a better resource is consumed. For example, if the program to find huge calculations of temperatures (Eq 5.4) is run on any individual machine (P3), it will take more time than the situation in which we run our program on the cluster nodes (P3). Let us consider a metallic rod initially at room temperature 20° C at all points, and temperatures are caused to change by suddenly cooling one end to 0°C and heating other end to 100°C. The program computes values until T > 80. Program generates huge computations in the form of matrix. If we run the program on one machine (p3) that is a part of the cluster, the execution time is larger as compared to running the program on other nodes of the cluster as well.



Temperature is found at interior points after each interval of time $t = dt$. The total number of points are N and total time is *tf*.

5.3 Program Definition Language (PDL)

Program Definition Language (PDL) also known as Pseudocode is a detailed yet readable description of what a computer program or algorithm must do, expressed in a formally styled natural language rather than in a programming language. Pseudocode is sometimes used as a detailed step in the process of developing a program. It allows designers or leads programmers to express the design in great detail and provides programmers a detailed template for the next step of writing code in a specific programming language.

Because pseudocode is detailed yet readable, the team of designers and programmers as a way to ensure can inspect it that actual programming is likely to match design specifications. Catching errors at the pseudocode stage is less costly than catching them later in the development process. Once the pseudocode is accepted, it is rewritten using the vocabulary and syntax of a programming language. Pseudocode is sometimes used in conjunction with CASE-based methodologies

5.3.1 Importance of PDL

Pseudocode is most useful after structure diagram has been done; it has a number of benefits or advantages, which are as follows:

- Using PDL helps the designers and programmers to understand the problem domain much easily.
- Writing PDL saves a large amount of time of the developers, during the construction & testing phase of a program's development.
- Allows programmers or designers to express design in plain language, which can be understood by all.
- While writing PDL the designer has the luxury of not bothering about the syntax of any language and can be precise about the exact problem.
- Through the pseudocode other programmers can also understand the exact methodology applied in designing that specific software, which becomes helpful in the maintenance of the software.

5.3.2 Procedural Design of Cluster Management System

The procedural design of the module CMS is as follows:

a. Initialize Cluster
Procedure: initialize cluster;

```
Interface Returns: initialize;
Begin
Type
        initialize_button IS BUTTON;

Begin   initialize_button check;
        IF button = 'Ok'
        THEN
                Accept server_node_name;
                Accept server_node_ip;
                Validate client_node_name;
                Accept client_node_name;
                Validate client_node_ip;
                Accept client_node_ip;
                Enable menus;

        END IF;

End initialize_button check;
End initialize cluster;
```

b. Shutdown Cluster

```
Procedure: shutdown cluster;
Interface Returns: None;
Begin
Type
Begin shutdown_cluster;
                IF cluster = initialized
                THEN
                 Delete Client Properties from files;
                 Disable Cluster;
                 Disable menus;
End shutdown_cluster;
End restart cluster;
```

c. Restart Cluster

```
Procedure: restart cluster;
Interface Returns: None;
Begin
Type
Begin cluster_restart;
                IF cluster = shutdown
                THEN
                Update Files;
                Enable Cluster;
                Enable keys;
```

```
End cluster_restart;
End restart_cluster;
```

d. Add Node

```
Procedure: Add Node;
Interface Returns: addnode;
Begin
Type
AddNodeDialog IS CLASS;
Begin
        From current dialog move to AddNodeDialog;
Begin add_node;
        Validate node_name;
        Validate node_ip;
        Update files;
        Update node_list;
End add_node;
End;
End Add_Node;
```

e. Delete Node

```
Procedure: Delete Node;
Interface Returns: deletenode;
Begin
Type
DeleteNodeDialog IS CLASS;
Begin
        From current dialog move to DeleteNodeDialog;
Begin delete_node;
        Validate node_selected;
        Update files;
        Update node_list;
End delete_node;
End;
End Delete Node;
```

f. Run IDE

```
Procedure: run_ide;
Interface Returns: None;
Begin
Type
        C_Ide IS IDE;
Begin
Show C_Ide;
End
End run_ide;
```

g. Compute Values

```
Procedure: compute_potential_values;
Interface Returns: computed potential values;
Begin
Type application data
Begin compute_values
        IF process= root
        THEN
                Display Initial Values
        ELSE
              Set V[I] = RATIO * ((U[I + 1] + ((1.0 - (2.0 * RATIO)) * U[I]));
              Set U[I] = RATIO * (V[I + 1] + V[I - 1]) + (1.0 – (2.0 *
                        RATIO))*V[I];
               Display V[I];
               Display U[I];
End compute_values;
End compute_potential_values;
```

5.3.3 Procedural Design of Integrated Development Environment

The procedural design of the module Integrated Development Environment is as follows:

a. File New

```
Procedure: File_New;
Interface Returns: None;
Begin
Type
        untitled.cpp IS FILE;
Begin new_file
        Set text_area = blank;
        Set file_name = 'untitled.cpp';
End new_file;
End File_New;
```

b. File Open

```
Procedure: File_Open;
Interface Returns: None;
Begin
Type
        file_chooser IS FILE CHOOSER
Begin open_file
        Open file_chooser;
        If file_name == valid
                Then
                        Open file;
```

```
                    Return;
                Else
                    Display Error in Status bar;
    End open_file;
    End File_Open;
```

c. File Save

```
Procedure: File_Save;
Interface Returns: None;
Begin
Type
        file_chooser IS FILE CHOOSER;
Begin save_file
        IF file already saved
        THEN
        Save file under current name;
        Display status in Status Bar;
        ELSE
        Open file_chooser;
        Save file under name specified;
        Display status in Status Bar;
End save_file;
End File_Save ;
```

d. File Save As

```
Procedure: File_SaveAs
Interface Returns: None;
Begin
Type
        file_chooser IS FILE CHOOSER;
        file_name IS STRING;
Begin saveas_file
        Open file_chooser;
        Save file under name specified;
        Display status in Status Bar;
End saveas_file;
End File_SaveAs;
```

e. Close File

```
Procedure: Close_File;
Interface Returns: None;
Begin
Type
Begin  file_close
        IF file == saved
        THEN
```

```
                Exit C_Ide;
        ELSE
                Call File_SaveAs;
                Exit C_Ide;
End file_close;
End Close_File;
```

f. Edit Cut

```
Procedure: Edit_Cut
Interface Returns: None
Begin
Type
Begin cut_edit
Cut text;
End cut_edit;
End Edit_Cut;
```

g. Edit Copy

```
Procedure: Edit_Copy
Interface Returns: None
Begin
Type
Begin copy_edit
Copy text;
End cut_edit;
End Edit_Copy;
```

h. Edit Paste

```
Procedure: Edit_Paste;
Interface Returns: None;
Begin
Type
Begin paste_edit
        IF text already cut or copy
        THEN
                Paste text;
End paste_edit;
End Edit_Paste;
```

i. Select All

```
Procedure: Select_All;
Interface Returns: None;
Begin
Type
Begin select_all_text
```

```
        IF text is present in text_area
        THEN
                Select all text;
                RETURN;
End select_all_text;
End Select_All;
```

j. Compile

```
Procedure: compile;
Interface Returns: Compilation Message
Begin
Type
Begin compile_file
        IF file == Saved
        THEN
                Compile_file;
                Display Compilation Message;
        ELSE
                Call File_Save;
                Compile_file;
                Display Compilation Message;
End compile_file;
End compile;
```

k. Run

```
Procedure: run;
Interface Returns: Output
Begin
Type
Begin run_file
        IF file == Compiled
        THEN
                Run file;
                Display Output;
        ELSE
                Call Compile_file;
                Run file;
                Display Output;
End run_file;
End run;
```

l. Properties

```
Procedure: node_properties;
Interface Returns: None;
Begin
Type
```

```
        number_node IS INTEGAR;
        npnumber IS INTEGAR;
Begin properties_node
        IF (number_node < 4 && number_node > 0)
                Set npnumber = number_node;
End properties_node;
End node_properties;
```

5.4 Technical Specifications

As the field of parallel development is relatively new in Pakistan, the need was felt to add a portion, which described the technical aspects of the platforms for which the software has been developed.

5.4.1 Full Specification

The following is a list of minimum system requirements for the software:

- CPU of i486 or above
- A network interface card that supports a TCP/IP stack
- An installed version of Linux Red Hat, preferably a *Red Hat 8.0.*
- Same Linux distribution and version on all nodes.
- Server / clients must have the same architecture (e.g., ia32 vs. ia64)
- Monitors and keyboards may be helpful, but are not required

CHAPTER 6

TESTING

6 Testing

Testing is done in actual to assure the quality of the product developed and it compares the actual functionality of the product developed with the initial requirements of the project. Testing can basically be defined as: once the source code has been generated then the software must be tested to uncover and correct as many errors or bugs in the program.

The biggest advantage of testing is that with its help it is possible to highlight the maximum number of bugs or errors in the program. No matter how well the developers develop the software or how well it is analyzed; there is always a chance of some glitches or bugs. Testing will help pinpoint those errors and later the programmer may remove them.

6.1 Methodology Adopted

The method and aims for testing are as follows:

- Developing test plans.
- Verifying software requirements.
- Performing internationalized software testing.
- Analyzing system response to load and stress.
- Analyzing application integration, regression and performance.

6.2 Methods of Testing

The final product was tested using the procedures of

1. Black Box and White box testing
2. Integration testing

For the Design and Implementation of CMS we have used Spiral Model in which on end of every phase analysis and testing takes place and on the basis of results the project steps into next phase. As per nature of our project we used Black Box Testing, Incremental Testing and Performance Testing to ensure the quality of our design.

6.2.1 Traceability Matrix

Reference	Test case Id	Transaction	Result
a	Test 1	Verify that the Cluster is initialized and the specific files are updated.	Pass
a	Test 2	Verify that after the Cluster is initialized, an alert is displayed.	Pass
a	Test 3	Verify that the menu items get enabled, if the Cluster is initialized.	Pass
b	Test 4	Verify that the Cluster shuts down	Pass
b	Test 5	Verify that the menu items get disabled.	Pass
c	Test 6	Verify that the Cluster gets restarted and the related menu enabled.	Pass
d	Test 7	Verify that the input by the user is in the correct format	Pass
d	Test 8	Verify that the list and the specific files are updated.	Pass
e	Test 9	Verify that the server node is not deleted, even if it is selected.	Pass
e	Test 10	Verify that the selected client node is deleted.	Pass
f	Test 11	Verify that the IDE is run.	Pass
a	Test 12	Verify that when user selects 'new', a new untitled file is opened.	Pass
b	Test 13	Verify that the File Chooser opens when	Pass

		the user selects 'open'	
b	Test 14	Verify that if the file does not exist, a message in the status bar is displayed.	Pass
c	Test 15	Verify that the "Save As" use case is called, if the work is being done on 'untitled.c'	Pass
c	Test 16	Verify that a message is displayed in the status bar, after the file is saved.	Pass
d	Test 17	Verify that the Save As dialog box opens, when the user invokes the Save As use case.	Pass
d	Test 18	Verify that the file is saved at the specified location.	Pass
d	Test 19	Verify that the message is displayed in the status bar.	Pass
e	Test 20	Verify that the selected text is cut.	Pass
f	Test 21	Verify that the Selected text is copied.	Pass
g	Test 22	Verify that the already cut/copied text is pasted.	Pass
h	Test 23	Verify that all the text is selected when select all is clicked.	Pass
i	Test 24	Verify that if no file is open, the 'File Open' use case is called.	Pass
i	Test 25	Verify that if the opened file is not	Pass

		saved, "File Save" use case is called.	
i	Test 26	Verify that after compilation, the 'Compiled Successfully' message appears in the output area.	Pass
i	Test 27	Verify that if the file has errors, the 'Compilation Errors' message is displayed in the output area.	Pass
j	Test 28	Verify that if no file is opened, the "File Open" use case if called.	Pass
j	Test 29	Verify that the file is run and the output is displayed in the output area.	Pass

Table 6.1: Traceability Matrix

6.2.2 Test Case Description

Test Case ID: Test 1	**Test Engineer:** Saira Junaid Sana Zubair Khan
Reference: a (Initialize Cluster)	
Test Date:	**Test Case Version:**
Objective:	Verify that the Cluster is initialized and the specific files are updated.
Product/Version/ Module	CMS Cluster Management System
Environment:	Linux Red Hat 8.0
Pre-Requisites	
Method: 1. Open the Cluster Management System. 2. Select *"Initialize Cluster"* from the options of the menu "Cluster Management". 3. Verify that after initialization, the files are updated.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 2	**Test Engineer:** Saira Junaid Sana Zubair Khan
Reference: a (Initialize Cluster)	
Test Date:	**Test Case Version:**
Objective:	Verify that after the Cluster is initialized, an alert is displayed.
Product/Version/ Module:	CMS Cluster Management System
Environment:	Linux Red Hat 8.0
Pre-Requisites:	
Method: 1. Open the Cluster Management System. 2. Select *"Initialize Cluster"* from the options of the menu "Cluster Management".	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 3	**Test Engineer:** Saira Junaid Sana Zubair Khan
Reference: a (Initialize Cluster)	
Test Date:	**Test Case Version:**
Objective:	Verify that the menu items get enabled, if the Cluster is initialized.
Product/Version/ Module:	CMS Cluster Management System
Environment:	Linux Red Hat 8.0
Pre-Requisites:	
Method: 1. Open the Cluster Management System. 2. Select *"Initialize Cluster"* from the options of the menu "Cluster Management".	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 4	**Test Engineer:** Saira Junaid Sana Zubair Khan
Reference: b (Shutdown Cluster)	
Test Date:	**Test Case Version:**
Objective:	Verify that the Cluster shuts down
Product/Version/ Module:	CMS Cluster Management System
Environment:	Linux Red Hat 8.0
Pre-Requisites:	The Cluster is initialized.
Method: 1. Open the Cluster Management System. 2. Select *"Shutdown Cluster"* from the options of the menu "Cluster Management". 3. Press the Shutdown Button.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 5	**Test Engineer:** Saira Junaid Sana Zubair Khan
Reference: b (Shutdown Cluster)	
Test Date:	**Test Case Version:**
Objective:	Verify that the menu items get disabled.
Product/Version/ Module:	CMS Cluster Management System
Environment:	Linux Red Hat 8.0
Pre-Requisites:	The Cluster is initialized.
Method: 1. Open the Cluster Management System. 2. Select *"Shutdown Cluster"* from the options of the menu "Cluster Management". 3. Press the Shutdown Button.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 6	**Test Engineer:** Saira Junaid Sana Zubair Khan
Reference: c (Restart Cluster)	
Test Date:	**Test Case Version:**
Objective:	Verify that the Cluster gets restarted and the related menus get enabled.
Product/Version/	CMS
Module:	Cluster Management System
Environment:	Linux Red Hat 8.0
Pre-Requisites:	The Cluster is not running.

Method:

1. Open the Cluster Management System.
2. Select *"Restart Cluster"* from the options of the menu "Cluster Management".
3. Press the "Restart" button.

Comments:

☑ **Passed** ☐ **Failed**

Test Case ID: Test 7	**Test Engineer:** Saira Junaid Sana Zubair Khan
Reference: d (Add Node)	
Test Date:	**Test Case Version:**
Objective:	Verify that the input by the user is in the correct format
Product/Version/	CMS
Module:	Cluster Management System
Environment:	Linux Red Hat 8.0
Pre-Requisites:	

Method:

1. Open the Cluster Management System.
2. Select *"Add Node"* from the options of the menu "Node Management".

Comments:

☑ **Passed** ☐ **Failed**

Test Case ID: Test 8	**Test Engineer**: Saira Junaid Sana Zubair Khan
Reference: d (Add Node)	
Test Date:	**Test Case Version:**
Objective:	Verify that the list and the specific files are updated.
Product/Version/ Module:	CMS Cluster Management System
Environment:	Linux Red Hat 8.0
Pre-Requisites:	The inputs are valid.
Method: **1.** Open the Cluster Management System. **2.** Select *"Add Node"* from the options of the menu "Node Management".	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 9	**Test Engineer**: Saira Junaid Sana Zubair Khan
Reference: e (Delete Node)	
Test Date:	**Test Case Version:**
Objective:	Verify that the server node is not deleted, even if it is selected.
Product/Version/ Module:	CMS Cluster Management System
Environment:	Linux Red Hat 8.0
Pre-Requisites:	
Method: **1.** Open the Cluster Management System. **2.** Select *"Delete Node"* from the options of the menu "Node Management".	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 10	**Test Engineer:** Saira Junaid Sana Zubair Khan
Reference e (Delete Node)	
Test Date	**Test Case Version**
Objective	Verify that the selected client node is deleted.
Product/Version/ Module	CMS CMS Integrated Development Environment
Environment	Linux Red Hat 8.0
Pre-Requisites	A Client Node is selected.
Method: **1.** Open the Cluster Management System. **2.** Select *"Delete Node"* from the options of the menu "Node Management". **3.** Select a Client Node.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 11	**Test Engineer** Saira Junaid Sana Zubair Khan
Reference f (Run IDE)	
Test Date	**Test Case Version**
Objective	Verify that the IDE runs.
Product/Version/ Module	CMS CMS Integrated Development Environment
Environment	Linux Red Hat 8.0
Pre-Requisites	The Cluster is running.
Method: **1.** Open the Cluster Management System. **2.** Select *"Run IDE"* from the options of the menu "Run Application". **3.** Press the "C/C++ IDE" button.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 12	**Test Engineer**: Saira Junaid Sana Zubair Khan
Reference: a (File New) **Test Date:**	**Test Case Version:**
Objective:	Verify that when user selects 'new', a new untitled file is opened.
Product/Version/ **Module:** **Environment:** **Pre-Requisites:**	CMS CMS Integrated Development Environment Linux Red Hat 8.0
Method: 1. Open the CMS Integrated Development Environment. 2. Select *"New"* from the options of the menu "File" or Click the *"New"* Button.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 13	**Test Engineer**: Saira Junaid Sana Zubair Khan
Reference: b (File Open) **Test Date:**	**Test Case Version:**
Objective:	Verify that the File Chooser opens when the user selects 'open'
Product/Version/ **Module:** **Environment:** **Pre-Requisites:**	CMS CMS Integrated Development Environment Linux Red Hat 8.0
Method: 1. Open the CMS Integrated Development Environment. 2. Select *"Open"* from the options of the menu "File" or Click the *"Open"* Button.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 14	**Test Engineer:** Saira Junaid Sana Zubair Khan
Reference: b (File Open)	
Test Date:	**Test Case Version:**
Objective:	Verify that if the file does not exist, a message in the status bar is displayed.
Product/Version/ Module:	CMS CMS Integrated Development Environment
Environment:	Linux Red Hat 8.0
Pre-Requisites:	
Method: 1. Open the CMS Integrated Development Environment. 2. Select *"Open"* from the options of the menu "File" or Click the *"Open"* Button.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 15	**Test Engineer:** Saira Junaid Sana Zubair Khan
Reference: c (File Save)	
Test Date:	**Test Case Version:**
Objective:	Verify that the "Save As" use case is called, if the work is being done on 'untitled.c'
Product/Version/ Module:	CMS CMS Integrated Development Environment
Environment:	Linux Red Hat 8.0
Pre-Requisites:	
Method: 1. Open the CMS Integrated Development Environment. 2. Select *"Save"* from the options of the menu "File" or Click the *"Save"* Button.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 16	**Test Engineer:** Saira Junaid Sana Zubair Khan
Reference: c (File Save)	
Test Date:	**Test Case Version:**
Objective:	Verify that a message is displayed in the status bar, after the file is saved.
Product/Version/ Module:	CMS CMS Integrated Development Environment
Environment:	Linux Red Hat 8.0
Pre-Requisites:	
Method: 1. Open the CMS Integrated Development Environment. 2. Select *"Save"* from the options of the menu "File" or Click the *"Save"* Button.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 17	**Test Engineer:** Saira Junaid Sana Zubair Khan
Reference: d (File Save As)	
Test Date:	**Test Case Version:**
Objective:	Verify that the Save As dialog box opens, when the user invokes the Save As use case.
Product/Version/ Module:	CMS CMS Integrated Development Environment
Environment:	Linux Red Hat 8.0
Pre-Requisites:	
Method: 1. Open the CMS Integrated Development Environment. 2. Select *"Save As"* from the options of the menu "File" or Click the *"Save As"* Button.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 18	**Test Engineer**: Saira Junaid Sana Zubair Khan
Reference: d (File Save As)	
Test Date:	**Test Case Version:**
Objective:	Verify that the file is saved at the specified location.
Product/Version/ Module:	CMS CMS Integrated Development Environment
Environment:	Linux Red Hat 8.0
Pre-Requisites:	The user has specified the file name and location.
Method: **1.** Open the CMS Integrated Development Environment. **2.** Select *"Save As"* from the options of the menu "File" or Click the *"Save As"* Button. **3.** Click the "Save" button on the "Save As" dialog box.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 19	**Test Engineer**: Saira Junaid Sana Zubair Khan
Reference: d (File Save As)	
Test Date:	**Test Case Version:**
Objective:	Verify that the message is displayed in the status bar.
Product/Version/ Module:	CMS CMS Integrated Development Environment
Environment:	Linux Red Hat 8.0
Pre-Requisites:	
Method: **1.** Open the CMS Integrated Development Environment. **2.** Select *"Save As"* from the options of the menu "File" or Click the *"Save As"* Button. **3.** Click the "Save" button on the "Save As" dialog	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 20	**Test Engineer**: Saira Junaid Sana Zubair Khan
Reference: e (Edit Cut)	
Test Date:	**Test Case Version:**
Objective:	Verify that the selected text is cut.
Product/Version/ Module:	CMS CMS Integrated Development Environment
Environment:	Linux Red Hat 8.0
Pre-Requisites:	Some text is selected.
Method: 1. Open the CMS Integrated Development Environment. 2.Select some text. 3. Select *"Cut"* from the options of the menu "Edit".	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 21	**Test Engineer**: Saira Junaid Sana Zubair Khan
Reference: f (Edit Copy)	
Test Date:	**Test Case Version:**
Objective:	Verify that the Selected text is copied.
Product/Version/ Module:	CMS CMS Integrated Development Environment
Environment:	Linux Red Hat 8.0
Pre-Requisites:	Some text is selected.
Method: 1. Open the CMS Integrated Development Environment. 2.Select some text. 3. Select *"Copy"* from the options of the menu "Edit".	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 22	**Test Engineer:** Saira Junaid Sana Zubair Khan
Reference: g (Edit Paste)	
Test Date:	**Test Case Version:**
Objective:	Verify that the already cut/copied text is pasted.
Product/Version/ Module:	CMS CMS Integrated Development Environment
Environment:	Linux Red Hat 8.0
Pre-Requisites:	Some text is already cut/copied.
Method: 1. Open the CMS Integrated Development Environment. 2.Select some text. 3. Select *"Paste"* from the options of the menu "Edit".	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 23	**Test Engineer:** Saira Junaid Sana Zubair Khan
Reference: h (Edit Select All)	
Test Date:	**Test Case Version:**
Objective:	Verify that all the text present in the text area is Selected.
Product/Version/ Module:	CMS CMS Integrated Development Environment
Environment:	Linux Red Hat 8.0
Pre-Requisites:	Some text is present in the text area.
Method: 1. Open the CMS Integrated Development Environment. 2.Select some text. 3. Select *"Select All"* from the options of the menu "Edit".	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 24	**Test Engineer:** Saira Junaid Sana Zubair Khan
Reference: i (Compile) **Test Date:**	**Test Case Version:**
Objective:	Verify that if no file is open, the 'File Open' use case is called.
Product/Version/ **Module:** **Environment:** **Pre-Requisites:**	CMS CMS Integrated Development Environment Linux Red Hat 8.0
Method: **1.** Open the CMS Integrated Development Environment. **2.** Select *"Compile"* from the options of the menu "Run" or press the "Compile" button.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 25	**Test Engineer:** Saira Junaid Sana Zubair Khan
Reference: i (Compile) **Test Date:**	**Test Case Version:**
Objective:	Verify that if the file is not saved, the "Save As" use case is called.
Product/Version/ **Module:** **Environment:** **Pre-Requisites:**	CMS CMS Integrated Development Environment Linux Red Hat 8.0
Method: **1.** Open the CMS Integrated Development Environment. **2.** Select *"Compile."* from the options of the menu "Run" or press the "Compile" button.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 26	**Test Engineer**: Saira Junaid Sana Zubair Khan
Reference: i (Compile)	
Test Date:	**Test Case Version:**
Objective:	Verify that after compilation, the 'Compiled Successfully' message appears in the output area.
Product/Version/ Module:	CMS CMS Integrated Development Environment
Environment:	Linux Red Hat 8.0
Pre-Requisites:	
Method: 1. Open the CMS Integrated Development Environment. 2. Select *"Compile"* from the options of the menu "Run" or press the "Compile" button.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 27	**Test Engineer**: Saira Junaid Sana Zubair Khan
Reference: i (Compile)	
Test Date:	**Test Case Version:**
Objective:	Verify that if the file has errors, the 'Compilation Errors' message is displayed in the output area.
Product/Version/ Module:	CMS CMS Integrated Development Environment
Environment:	Linux Red Hat 8.0
Pre-Requisites:	
Method: 1. Open the CMS Integrated Development Environment. 2. Select *"Compile"* from the options of the menu "Run" or press the "Compile" button.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 28	**Test Engineer**: Saira Junaid Sana Zubair Khan
Reference: j (Run)	
Test Date:	**Test Case Version:**
Objective:	Verify that if no file is opened, the "File Open" use case if called.
Product/Version/	CMS
Module:	CMS Integrated Development Environment
Environment:	Linux Red Hat 8.0
Pre-Requisites:	
Method: 1. Open the CMS Integrated Development Environment. 2. Select *"Run"* from the options of the menu "Run" or press the "Run" button.	
Comments:	
☑ **Passed**	☐ **Failed**

Test Case ID: Test 29	**Test Engineer**: Saira Junaid Sana Zubair Khan
Reference: j (Run)	
Test Date:	**Test Case Version:**
Objective:	Verify that the file is run and the output is displayed in the output area.
Product/Version/	CMS
Module:	CMS Integrated Development Environment
Environment:	Linux Red Hat 8.0
Pre-Requisites:	
Method: 1. Open the CMS Integrated Development Environment. 2. Select *"Run"* from the options of the menu "Run" or press the "Run" button.	
Comments:	
☑ **Passed**	☐ **Failed**

CHAPTER 7

CONCLUSION

7 Conclusion

At undergraduate level, doing a project of this magnitude was not an easy task, especially since development in the field of Parallel Computing is still new in Pakistan. But the successful completion of this project has given us the confidence and knowledge to work in the practical field as professional developers.

7.1 Comparison of the Final Project with the Project Objectives

To ensure that all the features, which were mentioned in the project proposal, have been fulfilled, and if any extra features have been added; a comparison has been done between the final project modules and their objectives.

a. **Cluster Management System**
   - **Implemented Features**
     The CMS has every feature as indicated in the project proposal.
   - **Unimplemented Features**
     None
   - **Additional Features**
     Node Addition/Deletion
     GUI-Based IDE

b. **CMS Integrated Development Environment**
   - **Implemented Features**
     The IDE has every feature as indicated in the project proposal.
   - **Unimplemented Features**
     None
   - **Additional Features**
     Edit Properties

7.2 Benefits for the User

This software provides the user with the following benefits:

- The CMS users may be able to perform complex calculations that previously were not possible on desktop computers.
- The Cluster Management System is an economical alternative for super computers/mainframes.
- The Cluster Management System provides the user a base line for projects requiring massive computations

- The IDE provides the user with a platform to run the Cluster Aware Applications in a GUI environment.

7.3 Good Features

This software has the following good features:

- Customization of nodes according to user requirements.
- Enhanced GUI interfaces.
- Effective implementation on Linux.
- Proper interaction with user.

7.4 Limitations

The CMS is only targeted for Linux because Cluster Management Software cannot be platform independent as different platforms have different requirements and compatibility issues. Another limitation is that the IDE has been developed only for parallel programs written in C.

7.5 Enhancement

Cluster computing is a rapidly maturing technology that seems certain to play an important part in the network centric computing future.

The IDE has been designed for running cluster aware applications in C. In future it can be enhanced for other languages like Java.

Presently in Pakistan cluster computing is a new dimension of parallel computing and a lot of research is yet to be done to develop on ground cluster computing environment. Keeping in view the above constraints, our objective in this field of cluster computing has been achieved up to some extent but still there are some loophole/deficiencies which could not be met due to time and other constraints or were out of the scope of our project. We hope that for future researches our project would be first step.

APPENDIX A

USER MANUAL

Appendix-A User Manual

The user manual is an important part of any documentation as it explains the interfaces of the software and helps the user to understand how to use the software. The user manual of the project "Cluster Management System" is as follows:

A.1 User Manual of Cluster Management System

The user manual of Cluster Management System shows the interface and the various situations, which may arise while using the software. The following are a few screen shots of the software.

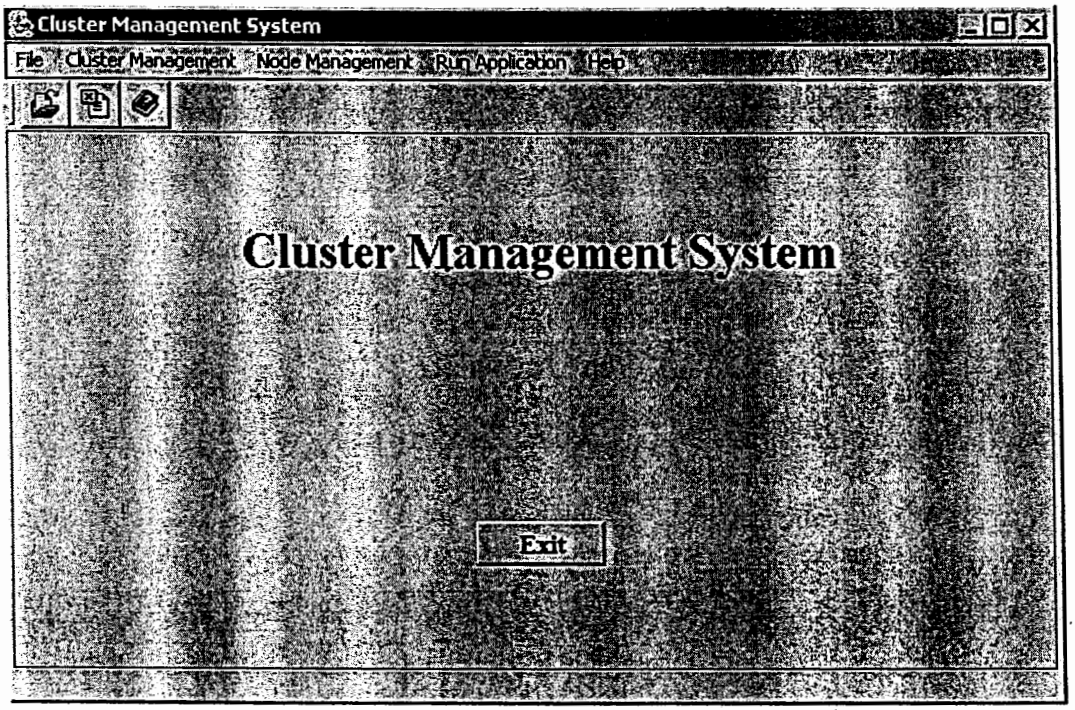


Fig A.1: CMS Main Interface

The 'Initialize Cluster' Dialog of the Cluster Management System is shown in figure A.2. Here, the Client Node Name and the IP Address are to be entered by the user, so as to initialize the cluster.

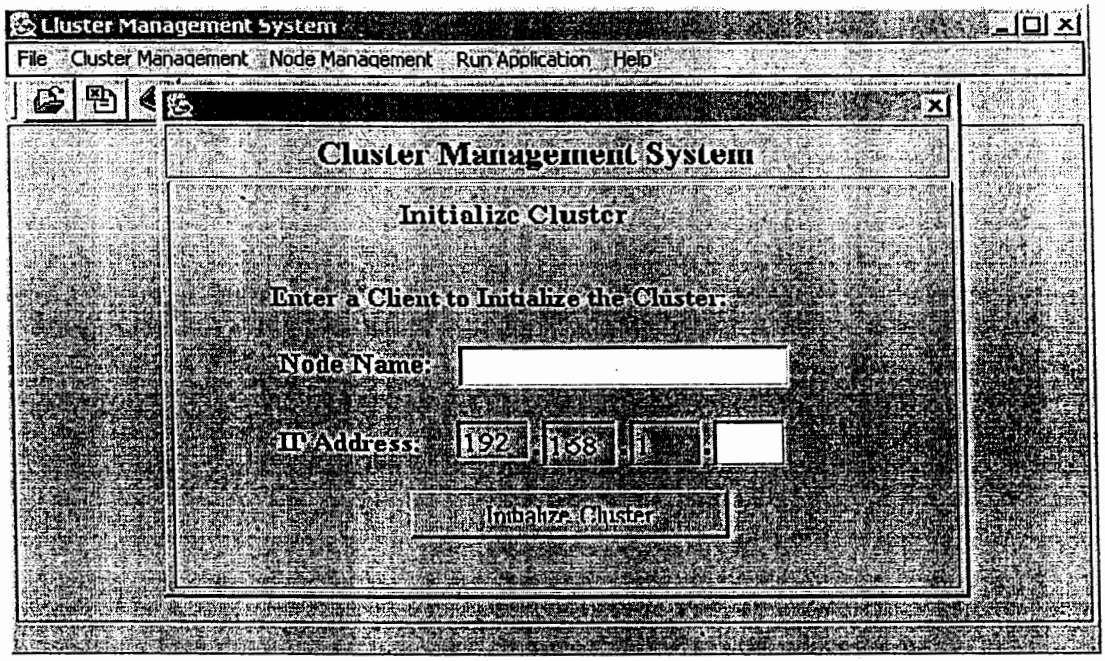


Fig A.2: Initialize Cluster Dialog

The Shutdown Cluster Dialog of the CMS is shown in figure A.3. It displays two options for the user; the user can either shutdown the cluster or cancel the option.

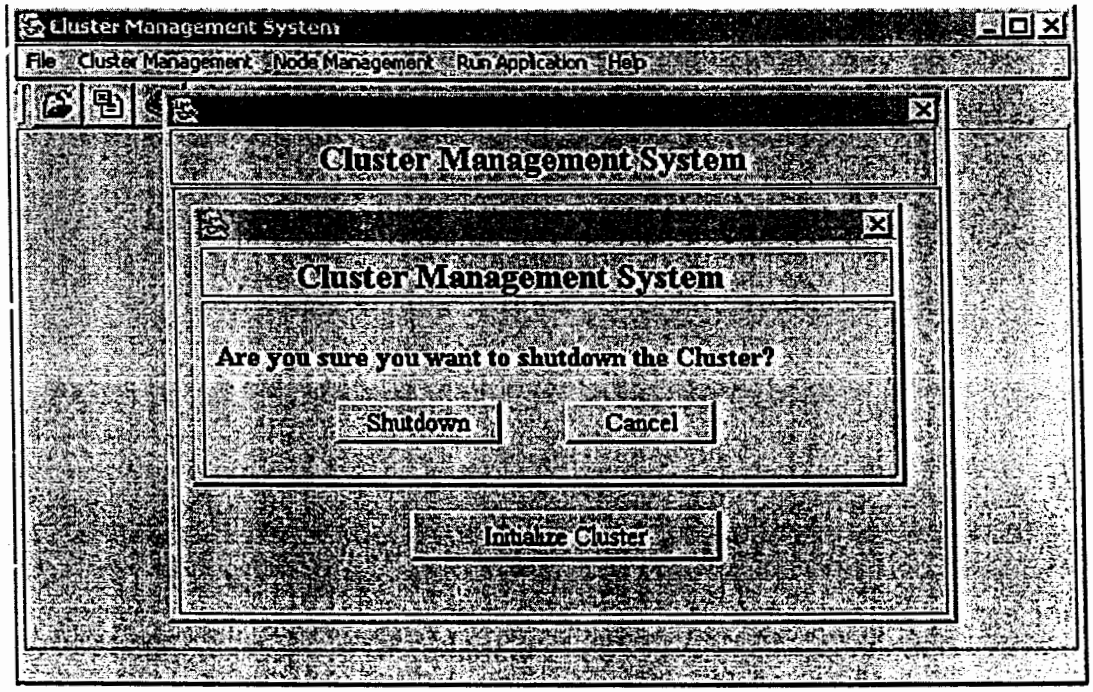


Fig A.3: Shutdown Cluster Dialog

The dialog box in figure A.4 provides the user with an option to restart the CMS, if previously, the CMS was shutdown.

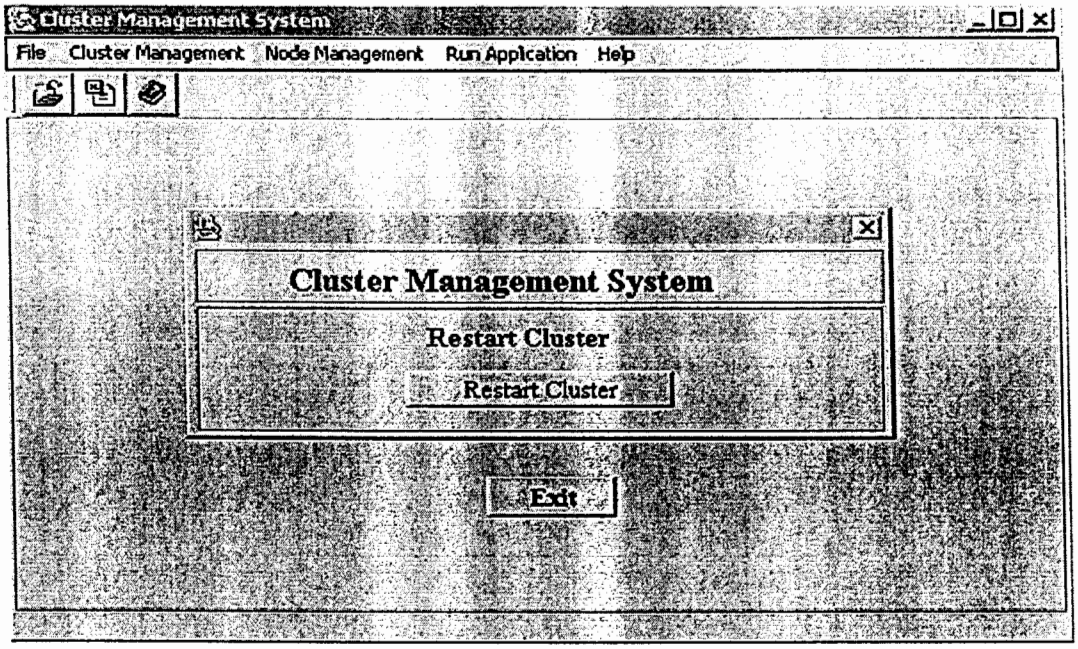


Fig A.4: Restart Cluster Dialog

The Add Node Dialog of the CMS is shown in figure A.5. It asks the user for the Node Name and the IP Address of the node to be added. It then verifies the input. If the input is valid, the changes are updated and the node is added.

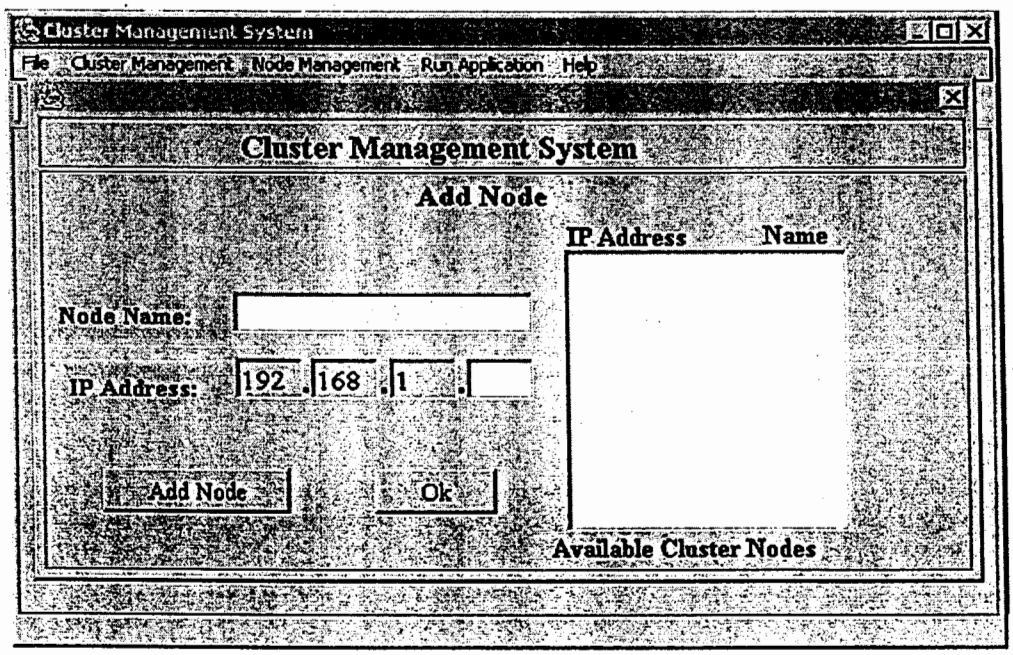


Fig A.5: Add Node Dialog

This dialog box in figure A.6 is used to delete the selected Node information from the system. Any slave node may be selected and then deleted, except the master node.

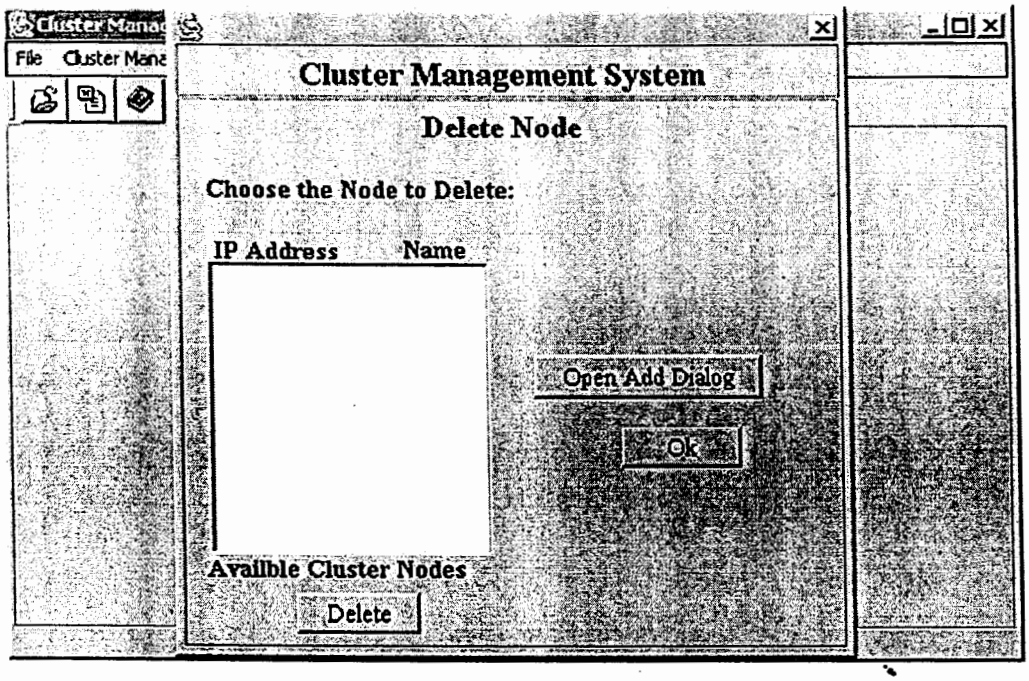


Fig A.6: Delete Node Dialog

Figure A.7 shows the Memory Monitor. It is used to monitor the memory load of the processor when the Cluster Management System executes a parallel application.

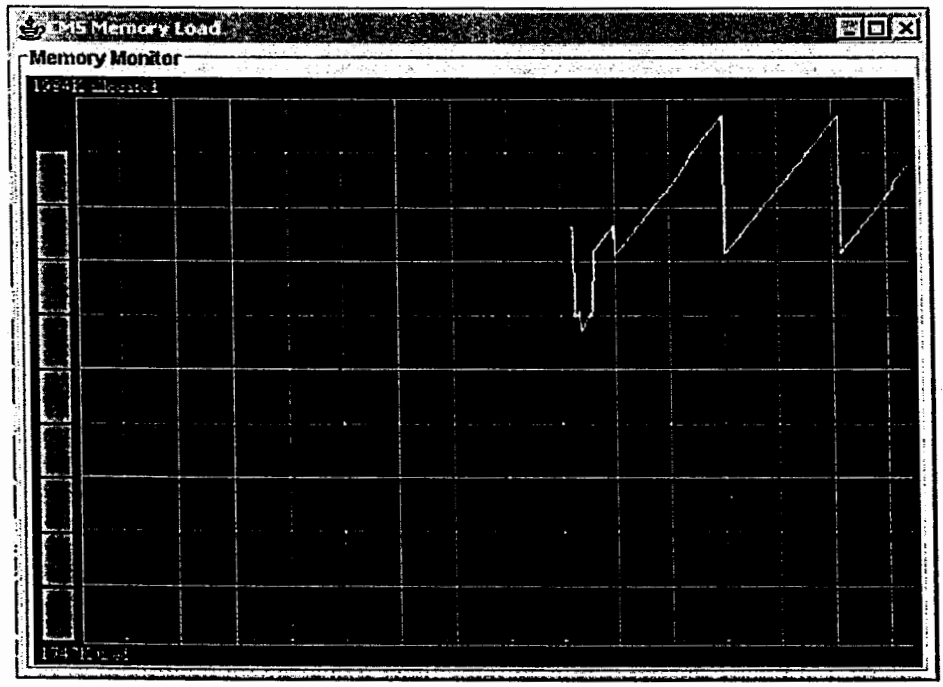


Fig A.7: Memory Monitor

The figure A.8 shows the Run IDE dialog box. It opens the Integrated Development Environment for parallel applications in C.

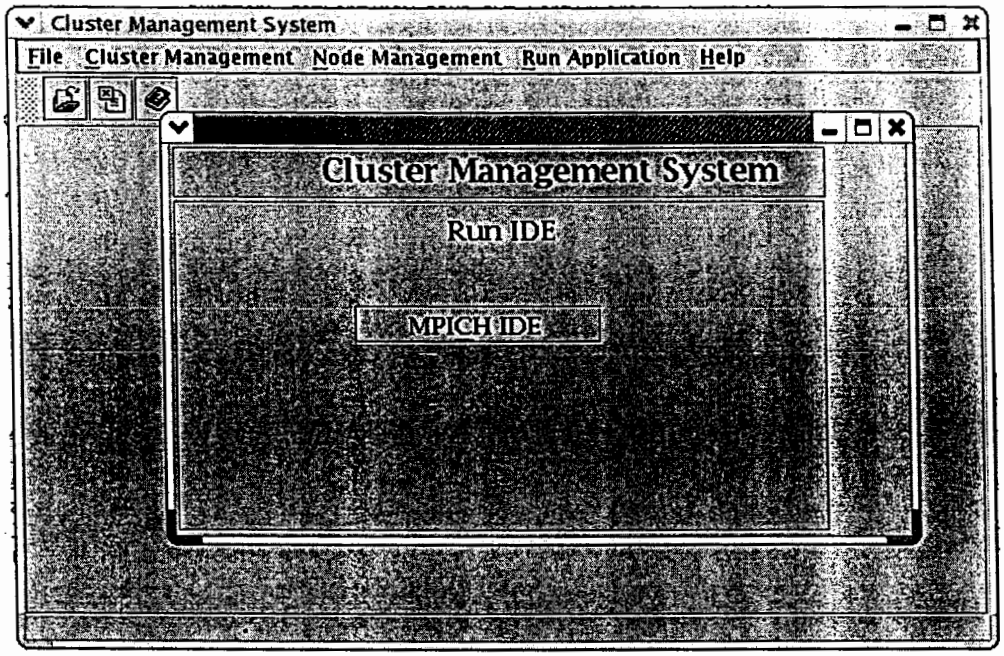


Fig A.8: Run IDE Dialog

A.2 User Manual of Integrated Development Environment

The figure A.9 shows the Integrated Development Environment interface, which the user comes across while developing and then executing the parallel applications in C.

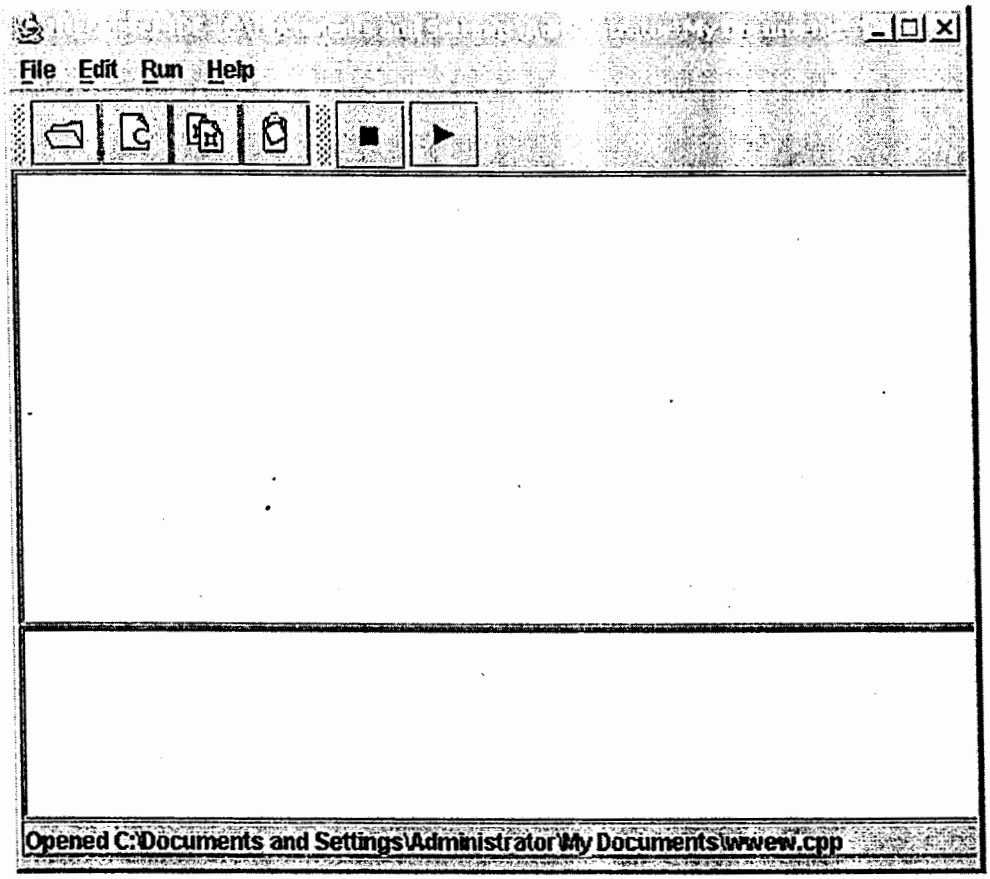


Fig A.9: IDE Interface

APPENDIX B

MPI PROGRAMMING

Appendix-B MPI Programming

The message passing interface libraries have been used in this project for development and execution of parallel programs. Following is a step by step procedure for installing and using the message passing interface.

B.1 Introduction

Message passing provides a portable means of implementing parallel programs across a wide variety of computers, from collections of trailing edge workstations (ORNL Stone Soup) to leading edge parallel computers. Portability is an important concern gives the short life time of current parallel computer systems. One of the most common message passing libraries is the Message Passing Interface (MPI).

This document gives a brief introduction to MPI. The following sections describe MPI in greater detail and illustrate its use with two simple examples written in Fortran and C.

B.2 Message Passing Interface

The Message Passing Interface (MPI) is a specification for a programming model. MPI consists of a library of functions and macros that can be used in C, C++, and Fortran programs. These functions implement communication operations among the multiple processors cooperating on a computation. The MPI model is a message passing model where processors at both the message source and the message destination must actively participate in the communication operation. MPI provides two types of communications:

1. Point-to-Point communication.
2. Collective communication.

MPI Point-to-Point communication operations send data from one processor to another processor. Each point-to-point communication operation involves a pair of processors; one processor sends the data and another processor receives it. For example, the send function (MPI_SEND) is the source of the message and the receive function (MPI_RECV) is the destination of the message.

MPI Collective communication is used to exchange data among several processors. Examples of collective operations are barriers synchronizations, reductions, and broadcasts.

B.3 Installing MPICH

The latest version of MPICH (UNIX all flavors) can be downloaded from www-unix.mcs.anl.gov/mpi/mpich/download.html to the master node. The installation instructions and other documents are also available on the site.

B.4 MPI Programming

The Message-Passing Interface (MPI) is a library of functions and macros that can be used in C, FORTRAN, C++ and JAVA programs. As its name implies, MPI is intended for use in programs that exploit the existence of multiple processors by message passing. MPI was developed in 1993-94 by a group of researchers from industry, government, and universities. As such, it is one of the first standards for programming parallel processors, and it is the first that is based on message passing. This appendix is a brief introduction to some important features of MPI.

B.5 General MPI Programs

Every MPI program must contain the preprocessor directive #include "mpi.h" This file, mpi.h, contains the definitions, macros and function prototypes necessary for compiling an MPI program. Before any other MPI functions can be called, the function MPI_Init must be called, and it should only be called once. Its arguments are pointers to the main function's parameters argc and argv. It allows systems to do any special set-up so that the MPI library can be used. After a program has finished using the MPI library, it must call MPI_Finalize. This cleans up any unfinished business" left by MPI e.g., pending receives that were never completed. So a typical MPI program has the following layout.

```
#include "mpi.h"
  main(int argc, char** argv)
    {
        MPI_Init(&argc, &argv);
        MPI_Finalize();
    }
```

MPI provides the function MPI_Comm_rank, which returns the rank of a process in its second argument. Its syntax is int MPI_Comm_rank (MPI_Comm comm, int rank). The first argument is a communicator. Essentially a communicator is a collection of processes that can send messages to each other. For basic programs, the only communicator needed is MPI_COMM_WORLD. It is predefined in MPI and consists of all the processes running when program execution begins.

Many of the constructs in programs also depend on the number of processes executing the program. So MPI provides the function MPI_Comm_size for determining this. Its first argument is a communicator. It returns the number of processes in a communicator in its second argument. Its syntax is

int MPI_Comm_size(MPI_Comm comm, int size)

The actual message passing in a program is carried out by the MPI functions MPI_Send and MPI_Recv. The first command sends a message to a designated process. The second receives a message from a process. These are the most basic message-passing commands in MPI. In order for the message to be successfully communicated the system must append some

information to the data that the application program wishes to transmit. This additional information forms the envelope of the message. In MPI it contains the following information.

- The rank of the Receiver.
- The rank of the Sender.
- A Tag.
- A Communicator.

These items can be used by the receiver to distinguish among incoming messages. The source argument can be used to distinguish messages received from different processes. The tag is a user-specified integer that can be used to distinguish messages received from a single process. For example, suppose process A is sending two messages to process B; both messages contain a single float. One of the floats is to be used in a calculation, while the other is to be printed. In order to determine which is which, A can use different tags for the two messages. If B uses the same two tags in the corresponding receives, when it receives the messages, it will know what to do with them. MPI guarantees that the integers 0-32767 can be used as tags. Most implementations allow much larger values.

As we noted above, a communicator is basically a collection of processes that can send messages to each other. When two processes are communicating using MPI_Send and MPI_Receive, its importance arises when separate modules of a program have been written independently of each other.

The syntax of MPI Send and MPI Receive is:

int **MPI_Send**(void* message, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)
int **MPI_Recv**(void* message, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status* status)

Like most functions in the standard C library most MPI functions return an integer error code. However, like most C programmers, we will ignore these return values in most cases. The contents of the message are stored in a block of memory referenced by the argument message. The next two arguments, count and data type, allow the system to identify the end of the message: it contains a sequence of count values, each having MPI type data type. This type is not a C type, although most of the predefined correspond to C types. The predefined MPI types and the corresponding C types are listed below:

MPI Data Type	C Data Type
MPI_CHAR	signed char
MPI_SHORT	signed short int
MPI_INT	signed int
MPI_LONG	signed long int
MPI_UNSIGNED	CHAR unsigned char
MPI_UNSIGNED_SHORT	unsigned short int
MPI_UNSIGNED	unsigned int

MPI_UNSIGNED_LONG	unsigned long int
MPI_FLOAT	float
MPI_DOUBLE	double
MPI_LONG_DOUBLE	long double
MPI_BYTE	
MPI_PACKED	

The last two types, MPI_BYTE and MPI_PACKED, don't correspond to standard C types. The MPI_BYTE type can be used if you wish to force the system to perform no conversion between different data representations (e.g., on a heterogeneous network of workstations using different representations of data).

The amount of space allocated for the receiving buffer does not have to match the exact amount of space in the message being received. For example, when our program is run, the size of the message that process sends, strlen (message)+1, is 28 chars, but process 0 receives the message in a buffer that has storage for 100 characters. In general, the receiving process may not know the exact size of the message being sent. So MPI allows a message to be received as long as there is sufficient storage allocated. If there isn't sufficient storage, an over flow error occurs. The arguments dest and source are, respectively, the ranks of the receiving and the sending processes. MPI allows source to be a \wildcard."

There is a predefined constant MPI_ANY_SOURCE that can be used if a process is ready to receive a message from any sending process rather than a particular sending process. There is not a wildcard for dest. As we noted earlier, MPI has two mechanisms specifically designed for partitioning the message space:" tags and communicators. The arguments tag and comm are, respectively, the tag and communicator. The tag is an int, and, our only communicator is MPI_COMM_WORLD, which, as we noted earlier is predefined on all MPI systems and consists of all the processes running when execution of the program begins. There is a wildcard, MPI_ANY_TAG that MPI_Recv can use for the tag. There is no wildcard for the communicator. In other words, in order for process A to send a message to process B; the argument comm that A uses in MPI Send must be identical to the argument that B uses in MPI_Recv. The last argument of MPI_Recv, status, returns information on the data that was actually received. It references a record with two fields one for the source and one for the tag. So if, for example, the source of the receive was MPI ANY SOURCE, then status will contain the rank of the process that sent the message.

B.6 Sample Program

This program makes use of multiple processes and makes each process send a greeting to another process. In MPI, the processes involved in the execution of a parallel program are identified by a sequence of non-negative integers. If there are p processes executing a program, they will have ranks 0, 1. . . p - 1. The following program has each process other than 0 send a message to process 0, and process 0 prints out the messages it received.

```
#include <stdio.h>
#include "mpi.h"
main (int argc, char** argv) {.0.
```

```
int my_rank;                    /* Rank of process */
int p;                          /* Number of processes */
int source;                     /* Rank of sender */
int dest;                       /* Rank of receiver */
int tag = 50;                   /* Tag for messages */
char message[100];              /* Storage for the message */
MPI_Status status;              /* Return status for receive */
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
MPI_Comm_size(MPI_COMM_WORLD, &p); if (my_rank != 0)
{
sprintf(message, "Greetings from process %d!", my_rank);
dest = 0;
MPI_Send(message, strlen(message)+1, MPI_CHAR, dest, tag,
MPI_COMM_WORLD);
}
else
{
        for (source = 1; source < p; source++)
{
        MPI_Recv(message, 100, MPI_CHAR, source, tag,
        MPI_COMM_WORLD, &status); printf("%s\n", message);
}
}
        MPI_Finalize();
}
```

B.7 Compiling and Running MPI Programs

To compile the C source program:
cc -o example1 example1.c $(MPI_INC) $(MPI_LIB)

To compile a FORTRAN program:
g77 -o example2 example2.f $(MPI_INC) $(MPI_LIB)

Where,

MPIR_HOME is where the MPI software is installed
MPI_INC is the location of the include files, e.g. mpi.h and mpif.h
MPI_LIB is the location of the precompiled software libraries

To run the program *example1* on *n* number of processors:
mpirun –np *n example1*

This will run the *example1* program on the number of processors mention at the place of *n*. The details of compiling and executing this program depend on the system being used.

When the sample program is compiled and run with two processes, the output is:

```
Greetings from process 1!
```

If it's run with four processes, the output is:

```
Greetings from process 1!
Greetings from process 2!
Greetings from process 3!
```

Although the details of what happens when the program is executed vary from machine to machine, the basics are the same on all machines:

i) The user issues a directive to the operating system which has the effect of placing a copy of the executable program on each processor.

ii) Each processor begins execution of its copy of the executable.

iii) Different processes can execute different statements by branching within the program. Typically the branching will be based on process ranks.

So, the Greetings program uses the Single Program Multiple Data (SPMD) paradigm i.e. we obtain the effect of different programs running on different processors by taking branches within a single program on the basis of process ranks. The statements executed by process 0 are different from those executed by the other processes, even though all processes are running the same program. This is the most commonly used method for writing MIMD programs.

APPENDIX C

GLOSSARY

Appendix-C Glossary

The glossary is the important part of the documentation as it defines all the major terms of the software and helps the user to understand the software. The glossary of the project is as follows.

Client

On a local area network or the internet, a computer that accesses shared network resources provided by another computer, called a server. A client facilitates a connection to server computers, and manages and presents information retrieved from those sources.

Cluster

A group of independent computer systems known as nodes or hosts, that works together as a single system to ensure that mission-critical applications and resources remain available to clients. The cluster nodes have neither monitor, nor keyboard, but they pass on additional computing power as well as memory.

Cluster Computing

A commonly found computing environment consists of many workstations connected together by a local area network.

Cluster Management Software

Cluster Management Software is a collection of fully integrated, easy to install software components designed to make it easy to build and use a cluster for high performance computing.

Communication Latency

The latency of a communication system is the minimum time taken to transmit one object, including any send and receive software overhead.

COTS

COTS (Commercial Off-The-Shelf) are commonly discussed as a requirement for parallel computing systems.

Heterogeneous

A heterogeneous architecture may be one in which some components are processors, and others memories, or it may be one that uses different types of processor together.

Homogeneous

A homogeneous architecture is one in which each element is of the same type - processor arrays and multi computers are usually homogeneous.

Integrated Development Environment:

An integrated development environment (IDE), also known as an integrated design environment and integrated debugging environment) is a computer software consisting of a text editor, a compiler, interpreter, or both, build-automation tools, and (usually) a debugger.

Latency

The time taken to service a request or deliver a message which is independent of the size or nature of the operation.

Message Passing

Message Passing is a style of inter-process communication in which processes send discrete messages to one another.

Message Passing Interface (MPI)

MPI not only unifies within a common framework programs written in a variety of existing (and currently incompatible) parallel languages but allows for future portability of programs between machines.

Node

Any single computer that is connected to a network is referred to as a node. A unique node number or address always differentiates a node from other nodes on the same network.

Parallel Job

This can be defined as a single application (job) that has multiple processes that run concurrently. Generally each process will run on a different processor (workstation) and communicate boundary, or other data, between the processes at regular intervals.

Parallel Processing

Parallel processing is a method of computation in which multiple processing modules operate in parallel, simultaneously controlling each other by sending signals back and forth. It is a processing in which multiple processors work on a single application simultaneously

Process

A process is a sequentially executing piece of code that runs on one processing unit of the system.

Sequential Computer

Synonymous with a Von Neumann architecture computer and is a "conventional" computer in which only one processing element works on a problem at a given time.

Sequential Job

This can de defined as a job that does not pass data to remote processes: typically such a job would run on a single workstation.

Server

A server is a computer, which provides some services to other computers referred to as clients.

Shared Memory

Shared memory is a model for interactions between processors within a parallel system. Systems physically share a single memory among their processors, so that any processor can directly access a value written to share memory by one processor.

Speedup

The ratio of two program execution times, particularly when the times are from execution on 1 and P nodes of the same computer. Speedup is usually discussed as a function of the number of processors, but is also a function (implicitly) of the problem size.

Supercomputer

A supercomputer is a time dependent term that refers to the class of most powerful computer systems world-wide at the time of reference.

BIBLIOGRAPHY AND REFERENCES

Bibliography and References

Books

1. Brawer, Steven; "Introduction to Parallel Programming"; Academic Press, San Diego; CA; 2002.
2. Deitel, Deitel; "Java How to Program"; Prentice Hall; 2002.
3. Khalid A. Mughal, Rolf W. Rasmussen;" A Programmer's Guide to Java Certification"; Addison-Wesley; 2001.
4. Craig Larman; "Applying UML and Patterns"; Prentice Hall; 2002.
5. Joseph Schmuller; "Sams Teach Yourself UML in 24 Hours"; Sams; 2001.
6. Roger S. Pressman; "Software Engineering - A Practitioner's Approach"; McGraw-Hill; 2001.

Websites

1. www.redhat.com
2. www.linuxplanet.com
3. www.linuxnetworx.com
4. www.linuxgazette.com
5. www.linuxfocus.org
6. www.beowulf.org
7. www.beowulf-underground.org
8. www.lam-mpi.org
9. www.mpi-forum.org
10. www.unix.mcs.anl.gov
11. www.metlin.org
12. www.dune.mcs.kent.edu
13. www.mrl.ucsb.edu
14. www.lam-mpi.org
15. www.cacr.caltech.edu
16. www.chinajavaworld.com
17. www.java.sun.com