
PODDAR INSTITUTE OF MANAGEMENT AND TECHNOLOGY
PODDAR
CALCUTTA

WEB ENABELED CLUSTER COMPUTATION WITH CUSTOMISED LOAD-BALANCING

A Report Submitted in Partial Fulfillment of the Requirement for the Degree of Bachelor of Technology

Submitted by

SUBHRENDU CHATTOPADHYAY
UNIVERSITY ROLL NO – 11501061032
REGISTRATION NO – 115010111022

SOUMADIP BISWAS
UNIVERSITY ROLL NO – 11501061021
REGISTRATION NO – 115010111013

JOYDIP DAS
UNIVERSITY ROLL NO – 11501061033
REGISTRATION NO – 115010111023

to the

Department of Computer Science and Engineering
B.P Poddar Institute of Management and Technology

May,2010

Certificate

This report of the Project titled

WEB ENABELED CLUSTER COMPUTATION WITH CUSTOMISED LOAD BALANCING

submitted by

Soumadip Biswas (Roll No.: 082209)
Subhrendu Chattopadhyay (Roll No.: 082213)
Joydip Das (Roll No.: 082215)

has been prepared under my supervision for the partial fulfillment of the requirements for B.Tech degree in Computer Science & Engineering under the West Bengal University of Technology
The report is hereby forwarded.

———————————————————-

Mr. Kalyan Mohanto
Asst. Prof. B.P.P.I.M.T
Department of CSE

countersigned by

———————————————————-

Mr. Somenath Roy Choudhury
Head of the Department B.P.P.I.M.T
Department of CSE

Acknowledgements

We would like to thank our supervisor MR. Kalyan Mohanta for his interest and support, who has been helping us throughout the year.

Abstract

The Beowulf cluster is a popular tightly coupled compute cluster designed for supercomputing work.

A wide variety of middle-ware like MPI (Message Passing Interface) or PVM (Parallel Virtual Machine) are moving in more advanced stage day by day.

In our project we design the user-friendly front-end by means of a web interface. Any cluster user can manage and monitor his computation and most important is that the user can take the help of the parallel algorithm deployment wizard to generate his program, no a-prior knowledge on parallel programming is required. So, that the cluster appear to the user as simple web application.

Contents

Chapter 1. Introduction

1.1 Introduction

A computer cluster is a group of linked computers, working together closely so that in many respects they form a single computer. The components of a cluster are commonly, but not always, connected to each other through fast local area networks. Clusters are usually deployed to improve performance and/or availability over that of a single computer, while typically being much more cost-effective than single computers of comparable speed or availability.

Clusters are mainly subset of Metacomputing which again subdivision of Distributed computing.

1.2 History

The formal *engineering* basis of cluster computing as a means of doing parallel work of any sort was arguably invented by Gene Amdahl of IBM, who in 1967 published what has come to be regarded as the seminal paper on parallel processing, Amdahl's Law. Amdahl's Law describes mathematically the speed-up one can expect from paralleling any given otherwise serially performed task on a parallel architecture. This article defined the engineering basis for both multiprocessor computing and cluster computing, where the primary differentiator is whether or not the interprocessor communications are supported "inside" the computer (on for example a customized internal communications bus or network) or "outside" the computer on a *commodity* network.

1. 3 Technologies

MPI is a widely-available communications library that enables parallel programs to be written in C, Fortran, Python, OCaml, and many other programming languages.

The GNU/Linux world supports various cluster software; for application clustering, there is Beowulf, Distcc, and MPICH. Linux Virtual Server, Linux-HA - director-based clusters that allow incoming requests for services to be distributed across multiple cluster nodes. MOSIX, OpenMosix, Kerrighed, OpenSSI are full-blown clusters integrated into the kernel that provide for automatic process migration among homogeneous nodes. OpenSSI, openMosix and Kerrighed are single-system image implementations.

1. 4 Classes of Cluster

According to the purpose or use of cluster computers they can be subdivided into many classes.

a. High Availability Cluster:-

Used for improving the availability of services that the cluster provides

b. Grid Computer:-

Grids are usually computer clusters, but more focused on throughput like a computing utility rather than running fewer, tightly-coupled jobs.

c. Load Balancing Cluster:-

When multiple computers are linked together to share computational workload or function as a single virtual computer.

d. Compute cluster:-

This type of cluster are used for emence computational jobs. This cluster design is usually referred to as Beowulf Cluster.

In our project we have concentrated on Beowulf Clusters or Compute Clusters. We have discussed Beowulf clusters, MetaComputing and MPI elaborately in later section.

Chapter 2. Preliminary

2.A Beowulf Cluster

2.A.1 Introduction

Originally referring to a specific computer built in 1994, Beowulf is a class of computer clusters similar to the original NASA system. They are high-performance parallel computing clusters of inexpensive personal computer hardware (commodity). The name comes from the main character in the Old English epic poem Beowulf.
Originally developed by Thomas Sterling and Donald Becker at NASA, Beowulf systems are now deployed worldwide, chiefly in support of scientific computing.

A Beowulf cluster is a group of usually identical PC computers running a Free and Open Source Software (FOSS) Unix-like operating system, such as BSD, Linux or Solaris. They are networked into a small TCP/IP LAN, and have libraries and programs installed which allow processing to be shared among them.

Beowulf is a concept whereby we use off the shelf computer components to assemble and build a cluster powerful enough to give us significant time and speed benefits when we use it to run computationally demanding programs like simulations, iterative algorithms etc. Unlike what is traditionally believed there is no hard and fast design rule for building a Beowulf cluster.

There is no particular piece of software that defines a cluster as a Beowulf. Commonly used parallel processing libraries include MPI (Message Passing Interface) and PVM (Parallel Virtual Machine). Both of these permit the programmer to divide a task among a group of networked computers, and collect the results of processing. An example of MPI software is Open Source Cluster Application Resources (OSCAR). OSCAR installs on top of a standard installation of a supported Linux distribution.

2.A.2 Types of Beowulf Cluster

As we have mentioned that Beowulf clusters are made of commodity hardware, so there are some classes of Beowulf clusters according to the choices of hardware components.

- **CLASS I BEOWULF**:-

This class of machines built entirely from commodity "off-the-shelf" parts.

Advantages:

➔Hardware is available from multiple sources (low prices, easy maintenance)
➔No reliance on a single hardware vendor
➔Driver support from Linux commodity
➔Usually based on standards (SCSI, Ethernet, etc.)

Disadvantages:

➔Best performance may require CLASS II hardware

- **CLASS II BEOWULF**:-

A CLASS II Beowulf is simply any machine that does not pass the Computer Shopper certification test. This is not a bad thing. Indeed, it is merely a classification of the machine.

Advantages:

➔Performance can be quite good!

Disadvantages:

➔Driver support may vary
➔Reliance on single hardware vendor
➔May be more expensive than CLASS I systems.

One CLASS is not necessarily better than the other. It all depends on your needs and budget. This classification system is only intended to make discussions about Beowulf systems a bit more succinct. The "System Design" section may help determine what kind of system is best suited for your needs.

2.B Fault Tolerance

2.B.1 Introduction

Fault-tolerance or graceful degradation is the property that enables a system to continue operating properly in the event of the failure of (or one or more faults within) some of its components.

Unfortunately, fault-tolerant computing is extremely hard, involving intricate algorithms for coping with the inherent complexity of the physical world. As it turns out, that world conspires against us and is constructed in such a way that, generally, it is simply not possible to devise absolutely foolproof, 100% reliable software. No matter how hard we try, there is always a possibility that something can go wrong. The best we can do is to reduce the probability of failure to an "acceptable" level. Unfortunately, the more we strive to reduce this probability, the higher the cost.

Fault Tolerance Points in static cluster

- When any node gets offline, the load of the load should be distributed. This can be done by picking a single node from available free nodes, or can further be distributed among available no of nodes during runtime.
- When the process starts one can find no of available nodes and then distribute load accordingly assuming the no would remain same.

Fault Tolerance Points in dynamic cluster

- When using P2P model, any node can go up or down at any time, so we can separate the calculation which are incomplete sue to node break down and then redistribute them among newly available node in the system, or if there is no new node available , we can wait until any free node available.

There can be many techniques that can be used for these purpose.

Fault Tolerance Points customized model

In case of customized model user can be asked to choose any fault tolerance model available.

2.C Load Balancing

2.C.1.Introduction

In computer networking, load balancing is a technique to distribute workload evenly across two or more computers, network links, CPUs, hard drives, or other resources, in order to get optimal resource utilization, maximize throughput, minimize response time, and avoid overload. Using multiple components with load balancing, instead of a single component, may increase reliability through redundancy.

It is commonly used to mediate internal communications in computer clusters, especially high-availability clusters. If the load is more on a server, then the secondary server takes some load while the other is still processing requests.

2.C.2.Load balancer features

Hardware and software load balancers can come with a variety of special features.

- Asymmetric load: A ratio can be manually assigned to cause some back end servers to get a greater share of the workload than others. This is sometimes used as a crude way to account for some servers being faster than others.
- Priority activation: When the number of available servers drops below a certain number, or load gets too high, standby servers can be brought on-line.
- SSL Offload and Acceleration: SSL applications can be a heavy burden on the resources of a Web Server, especially on the CPU and the end users may see a slow response (or at the very least the servers are spending a lot of cycles doing things they weren't designed to do). To resolve these kinds of issues, a Load Balancer capable of handling SSL Offloading in specialized hardware may be used. When Load Balancers are taking the SSL connections, the burden on the Web Servers is reduced and performance will not degrade for the end users.
- Distributed Denial of Service (DDoS) attack protection: load balancers can provide features such as SYN cookies and delayed-binding (the back-end servers don't see the client until it finishes its TCP handshake) to mitigate SYN flood attacks and generally offload work from the servers to a more efficient platform.

- TCP buffering: the load balancer can buffer responses from the server and spoon-feed the data out to slow clients, allowing the server to move on to other tasks.
- Direct Server Return: an option for asymmetrical load distribution, where request and reply have different network paths.
- Health checking: the balancer will poll servers for application layer health and remove failed servers from the pool.
- HTTP caching: the load balancer can store static content so that some requests can be handled without contacting the web servers.
- Content Filtering: some load balancers can arbitrarily modify traffic on the way through.
- Priority queuing: also known as rate shaping, the ability to give different priority to different traffic.
- Content aware switching: most load balancers can send requests to different servers based on the URL being requested.
- Client authentication: authenticate users against a variety of authentication sources before allowing them access to a website.
- Programmatic traffic manipulation: at least one load balancer allows the use of a scripting language to allow custom load balancing methods, arbitrary traffic manipulations, and more.
- Firewall: direct connections to back end servers are prevented, for network security reasons
- Intrusion Prevention System: offer application layer security in addition to network/transport layer offered by firewall security.

2.C.3 Load balancing

Load balancing is a very useful mechanism, for systems uses multiple nodes for execution. I our project it can be very helpful to fulfil our objectives of getting efficient execution.

We can incorporate this load balancing scenario in our project in many different ways,like :

- Balance the load within available no of nodes assuming no of nodes are static
- Again, if the cluster is a dynamic type then we can use the load balancing calculation before the execution starts by calculating no of available nodes at that time

- Load balancing can be done by determining the efficiency of different nodes available and then distributing the load accordingly
- Another way of balancing the load is to give the users the privilege of customising the load themselves over the nodes, thus providing much more flexibility

Next is the concept of peer to peer (P2P) computing in load balancing, when one user is using your service from his/her machine, their m/c is connected to the server and can work as a node at the time of receiving the service, thus can provide more efficient service to all users.

2.D Meta-Computation

2.D.1 Introduction

Meta computing is all computing and computing-oriented activity which involves computing knowledge (science and technology) common for the research, development and application of different types of computing. It may also deal with numerous domains of computing application, such as: industry, business, management, as well as human.

New emerging field of meta computing is focused on the methodological and technological aspects of the development of large computer networks/grids, such as Internet, intranet and other territorially distributed computer networks for special purposes.

So actually the meta computer is, simply put, a collection of computers held together by state-of-the-art technology and "balanced" so that, to the individual user, it looks and acts like a single computer. The constituent parts of the resulting "meta computer" could be housed locally, or distributed between buildings, even continents.

2.D.2 History

The term "meta computing" was coined around 1987 by NCSA Director, Larry Smarr. But the genesis of meta computing at NCSA took place years earlier, when the center was founded in 1986. Smarr's goal was to provide the research community with a "Seamless Web" linking the user interface on the workstation and supercomputers.

By 1988, NCSA's vision of the meta computer, as far as hardware was concerned, consisted of vector multi-processors integrated with the newly-emerging massively parallel architectures.

Between 1990 and 1995, in addition to exploiting novel computing architectures, NCSA also participated in a number of significant testbeds and demonstration projects, each designed to probe the technologies needed for effective meta computing.

2.D.2 Road Blocks

NCSA and other supercomputing centers have already shown the potential of meta computing on both local and, in a limited sense, a national scale. But many hurdles remain before meta computing is no longer just a grand concept but becomes an everyday reality.

- Because configurations of meta computers on a national scale will more likely be heterogeneous, there is an urgent need to develop more robust, "universal" computer languages like Message Passing Interface. Then there's the the problem of optimizing the use of diverse machines for a given problem. Compilers must be smart enough to recognize which parts of a program are best tackled by distinct sets of processors.

- The national meta computer demands faster networks with more access points. The speed with which data moves over wide-area networks is but a fraction of local area networks; local area networks. High-speed networks must be available not only to a few supercomputing centers but to every school, business and home.

- Meta computing demands superior user interfaces enabling easy, rapid access to diverse computational resources in a fully "transparent" manner; that is the user need have no knowledge of the myriad "widgets" in hardware and software lying behind the interfaces. Today, the World Web Web and the multitude of browsers it has spawned, particularly the newer, more interactive browsers such as "Hot Java" , could provide a foundation for such interfaces to emerge.

2.E MPI <Message Passing Interface>

2.E.1 Introduction

Message Passing Interface (MPI) is a specification for an API that allows many computers to communicate with one another. It is used in computer clusters and supercomputers.

MPI is a language-independent communications protocol used to program parallel computers. Both point-to-point and collective communication are supported. MPI "is a message-passing application programmer interface".

There are mainly three version of MPI are available.

MPI-1.1 , MPI-1.2 and MPI-2.0

2.E.2 Reasons for using MPI

Standardization - MPI is the only message passing library which can be considered a standard. It is supported on virtually all HPC platforms. Practically, it has replaced all previous message passing libraries.

Portability - There is no need to modify your source code when you port your application to a different platform that supports (and is compliant with) the MPI standard.

Performance Opportunities - Vendor implementations should be able to exploit native hardware features to optimize performance.

Functionality - Over 115 routines are defined in MPI-1 alone.

Availability - A variety of implementations are available, both vendor and public domain.

2.E.3 History

MPI resulted from the efforts of numerous individuals and groups over the course of a 2 year period between 1992 and 1994. Some history: –

- 1980s - early 1990s: Distributed memory, parallel computing develops, as do a number of incompatible software tools for writing such programs - usually with tradeoffs between portability, performance, functionality and price. Recognition of the need for a standard arose.

- April, 1992: Workshop on Standards for Message Passing in a Distributed Memory Environment, sponsored by the Center for Research on Parallel Computing, Williamsburg, Virginia. The basic features essential to a standard message passing interface were discussed, and a working group established to continue the standardization process. Preliminary draft proposal developed subsequently.

- November 1992: - Working group meets in Minneapolis. MPI draft proposal (MPI1) from ORNL presented. Group adopts procedures and organization to form the MPI Forum. MPIF eventually comprised of about 175 individuals from 40 organizations including parallel computer vendors, software writers, academia and application scientists.

- November 1993: Supercomputing 93 conference - draft MPI standard presented.

- Final version of draft released in May, 1994 - available on the at: http://www-unix.mcs.anl.gov/mpi.

- MPI-2 picked up where the first MPI specification left off, and addressed topics which go beyond the first MPI specification. The original MPI then became known as MPI-1. MPI-2 is briefly covered later. Was finalized in 1996.

- Today, MPI implementations are a combination of MPI-1 and MPI-2. A few implementations include the full functionality of both.

Chapter 3. Previous Works

3.1 Introduction

There are many High Performance Computer Models. These High-performance computing (HPC) uses supercomputers and computer clusters to solve advanced computation problems. But Beowulf clusters are mostly popular for its cost efficiency. In this section we are going to mention some of HPC projects which are build on Beowulf cluster.

3.2 Beowulf Cluster Projects

A. IOFOR

Iofor was one of the first Beowulf system, built very cheaply in early 1998 using second-hand 486 PCs. It was used for experiments, software development and testing, and research into parallel I/O and data tiling.

B. PERSEUS

Perseus is one of first purpose-built production system, developed for computational chemistry applications. Officially known as the South Australian Computational Chemistry Facility, the machine consists of 116 dual-Pentium PCs (i.e. 232 processors) connected by 100 Mbit/s switched Ethernet, with a peak speed of 113 GFlops. When commissioned in March 2000 it was the largest and fastest cluster in Australia and one of the largest PC clusters in the world. It is dedicated to computational chemistry problems using the electronic structure applications GAUSSIAN and GAMESS.

Used configuration in PERSEUS

Hardware

- Pentium II processors
- 256 MB of RAM
- Gigabit Ethernet for networking

Software

- Linux X_86 as Operating System <RHEL>
- CONDOR a cluster management software
- MPICH ,LAM/MPI and PVM as message passing interface

C. ORION

Orion is a commercial cluster from Sun Microsystems, used by physicists studying the fundamental forces of nature and the basic structure of matter (quarks, gluons, leptons, etc). Officially known as the National Computing Facility for Lattice Gauge Theory, the cluster is a Sun Technical Compute Farm consisting of 40 4-way E420R nodes (i.e. 160 processors) connected by a high-speed Myrinet network. It was the fastest computer in Australia when installed in June 2000, and ranked #188 in the November 2000 list of the Top 500 supercomputers in the world. It has shown peak performance of 110 GFLops in Linpack bench-marking tool.

Used configuration in ORION

Hardware

- 40 4 way ultra-sparc II 450 Ghz processors
- Total 160 GB of RAM
- 720 GB of disk
- 640 MB of cache
- 100 Mbit/s fast Ethernet

Software

- Solaris Operating System
- Sun HPC Cluster Tools
- Sun Performance Library
- Sun Scalable Scientific Subroutine Library
- PGHPF compiler

D. HYDRA

Hydra is used for a variety of applications including computational physics, chemistry, biotechnology, engineering, geoscience, petroleum

engineering, applied mathematics, water resource management, and computational fluid dynamics.

Hydra is an IBM e Server 1350 Linux cluster with a peak performance of 1.2 T Flops. It was the second fastest computer in Australia when installed in June 2003. It was ranked 106th in the June 2003 list of the top 500 fastest supercomputers in the world, with a measured Linpack benchmark result of 840 G Flops.

Used system in HYDRA

Hardware

- 129 Intel dual Xeon Processors 2.4 Ghz
- 2 GB of RAM in each processor
- Myrinet 2000 optical fibre network

Software

- Red Hat Enterprise Linux 3 - operating system.
- Torque - queuing system and scheduler.
- Portland Group Cluster Development Kit - including compilers, a debugger (pgdbg) and a profiler pgprof.
- Intel Compiler Suite - including C, C++, F77 and F95 compilers.
- MPICH-GM - library for MPI message-passing parallel programming over Myrinet.
- MPICH - library for MPI message-passing parallel programming over ethernet.
- Intel MKL - Intel's Math Kernel Library
- BLAS, LAPACK and ScaLAPACK
- ARPACK
- FFTW
- SPRNG
- SuperLU
- NetCDF

E. AQUILA

Aquila was purchased by SAPAC <*The South Australian Partnership for Advanced Computing* >with funding from a number of research groups and a contribution of AU$1.035 million from the State Government of South Australia.

Aquila is used for a variety of applications in the computational sciences and is particularly useful for problems with extremely large data sets or that require a large memory footprint.

Used configuration in AQUILA

Hardware

- 160 Itanium2 processors with 1.3 Ghz
- Total of 160 GB RAM, may be shared by each processor or used by a single processor.
- 3 MB of Cache for each processor

Software

- Red Hat Enterprise Linux
- Torque - queuing system and scheduler.
- Intel Compiler Suite - including C, C++, F77 and F95 compilers.
- SGI MPT - Message Passing Toolkit - Libraries for parallel programming on SGI architecture.

3.3 Web Based Projects on Cluster

A. wCLUSTO

It is a web enabled clustering toolkit used for information gathering from gnomic data samples. It is a web enabled version of a stand alone software CLUSTO. This system it brain child of

Matthew D. Rasmussen, Mukund S. Deshpande, George Karypis,
James Johnson, John A. Crow, and Ernest F. Retzel
of
Department of Computer Science and Engineering,
University of Minnesota

B. PARASITE

Computing potential is wasted and idle not only when applications are executed, but also when a user navigates by Internet. To take advantage of this, an architecture named Parasite has been designed in order to use distributed and networked resources, without disturbing local computation. This is the working principle of certain global computing projects, but this development introduces new ideas with respect to user intervention, the distributed programming paradigm or the resident

software on each user's computer. The project is based on developing software technologies and infrastructures in order to facilitate Web-based distributed computing. This paper outlines the most recent advances in the project, as well as discussing the architecture developed and an experimental framework that would validate this infrastructure.

This paper dealt with another potential technology called PARASITE computation, which is very effective but not implemented because of several reasons.

This idea was proposed by
Remo Suppi, Marc Solsona, Emilio Luque
in their paper “Web-Based Distributed Computing Using Parasite ”
published in IEEEcomputersociety
Genova, Italy
February 05-February 07

C. ALCHEMI

Alchemi is an open source software framework that allows you to painlessly aggregate the computing power of networked machines into a virtual supercomputer (desktop grid) and to develop applications to run on the grid. It is a project developed by The University Of Manchester.

It has been designed with the primary goal of being easy to use without sacrificing power and flexibility.

Alchemi includes:

- The runtime machinery (Windows executables) to construct computational grids.
- A .NET API and tools to develop .NET grid applications and grid-enable legacy applications.

D. OSCAR

OSCAR allows users, regardless of their experience level with a *nix environment, to install a Beowulf type high performance computing cluster. It also contains everything needed to administer and program this type of HPC cluster. OSCAR's flexible package management system has a rich set of pre-packaged applications and utilities which means you can get up and running without laboriously installing and configuring complex cluster administration and communication packages. It also lets administrators create customized packages for any kind of distributed application or utility, and to distribute those packages from an online package repository, either on or off site.

OSCAR installs on top of a standard installation of a supported Linux distribution. It installs and configures all required software for the selected packages according to user input. Then it creates customized disk images which are used to provision the computational nodes in the cluster with all the client software and administrative tools needed for immediate use. OSCAR also includes a robust and extensible testing architecture, ensuring that the cluster setup you have chosen is ready for production.

E.CONDOR

Condor is a specialized workload management system for compute-intensive jobs. Like other full-featured batch systems, Condor provides a job queuing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. Users submit their serial or parallel jobs to Condor, Condor places them into a queue, chooses when and where to run the jobs based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion.

Chapter 4. Objective

In modern days the need of computation power has become an obvious thing. Because many computer softwares are so complex that they need a cumbersome amount of processing speed along with other resources. Also the domain of computer has become larger than ever before. Every person from an engineer to common people need computers. This has become possible for only reason that a computer can do a lot of works with some easy interfaces. But it is not true for all cases. We can site some situation in support of this logic.

Suppose a mathematician need to compute some heavy calculation say sum of square-root of 1000000 natural numbers. In this situation the person need to spend a lot amount of time or he may go to some cluster computation center. Then he have to acquire knowledge of some programming language in which he can frame his problem in that cluster.

So to avoid this type of situation we have decided to configure a system which can help a common user to frame his huge computational job to submit over Internet and get his result. A more sophisticated user like C programmers have opportunity frame their problem himself. But instead of going to a cluster computation center he can submit his job to the cluster via web interfaces.

In our model of implementation of cluster we have also given a stress on heterogeneity of the system and cost effectiveness of the project. We have also given sufficient importance to make the user interface as simple as possible. We have also point out to the points where one can improve the idea in this document.

In future this type of computation solution (called meta-computation) will be everywhere. Here are some of the futuristic view of situation which can be achieved by this type of tools.

Like, Classroom computers could open a window to high performance computing resources. Students may learn about tornadoes by running a simulation on a supercomputer hundreds of miles away and seeing the results displayed on a classroom screen moments later. In scaled-down virtual environments, they'll be able to compute and walk around molecules, or immerse themselves in three-dimensional mathematical objects which can be rotated, distorted, or compacted at will.

Chapter 5. Feasibility Study

5.1 Introduction

Every proposal must have some feasibility study behind. If a proposal is not feasible then proposal is not suitable to be adopted.

In our project we have done extensive study to check its feasibility. We have shown all the feasibility criteria in this part of the documentation.

There are mainly three types of feasibility study.

1.Economic Feasibility
2.Technical Feasibility
3.Operational Feasibility

5.2 Economic Feasibility

Economic feasibility concerns with return from investment in a project. It was a bit difficult to judge the economic feasibility of the project. This is because it was not straightforward estimated hardware costs for the proposed system in isolation with other system that were under the process of development. New hardware and software installation cost for the installing the project will be incurred but user department can effort the cost incurred by the manual system and service render. Considering the other point of economical feasibility, it was estimated that although the organization will be spending more money for computer but the overall expenditure would be much lesser that of a pen and paper system. Moreover, it is a industry and as
all the employee have to pay sufficient amount of money, the administration feel the necessity of computerization to provide better service to the employee and agree to pay necessary money for computerization which is turn gives intangible benefit to the industry so the project is found to be economically feasible.

This project is need to have at least two commodity computer along with Linux operating system which can be available freely. It also needs LAN connectivity between them along with a web server with php support (preferably apache HTTP server). So these things are not so much costly. More over the throughput of the system will be nearly twice, which makes the system economically feasible.

5.3 Technical Feasibility

As a user goes for computerization for the time and show it has purchased a PC Pentium III processor, Linux as operating system. In context of the given technical environment the proposed system will be developed in C binding of LAM/MPI with a php web server support which is well supported by the above mentioned environment. Though the user system does not need to have technically skilled person for handling the computers but they will be given different level of training depending on types of services before the system is in full operation. It will not to be a problem because people carrying the manual system are well trained. Since the system will be user friendly with on line help at all steps, high degree of expertise will not be necessary here. Hence, we conclude that the project is technically feasible.

Chapter 6. System Requirement

6.1 Introduction

At least two computers needed in these project each of them must posses each of the hardware and software System Requirements.

6.2 Hardware Requirement

- Any processor
- 10 GB Hard Disk Free Space
- 256 MB RAM
- Ethernet cable
- Switch

6.3 Software Requirement

FOR SERVER MACHINE

- Linux Operating System
- gcc Compiler
- LAM/MPI with C/C++ Binding
- SSH
- Apache HTTP Server
- php for server side scripting

FOR CLIENT MACHINE

- Any Operating System
- Standard JavaScript Supported Browser

Language used for this Project is C with LAM/MPI Wrapper compiler. And for server side scripting we have used php 5. All the software s used are free and open source they can be easily achieved.

6.4 System we have used

HARDWARE

- Intel Pentium IV processor
- 10 GB Hard Disk Free Space
- 256 MB RAM
- Ethernet cable
- Switch

SOFTWARE

<For server>

- Mandriva Spring 2007 Operating system
- gcc Compiler version 4.1.2
- LAM/MPI with C/C++ Binding 7.1.4
- SSH 4.6p1
- Apache HTTP Server 2.2.4
- php 5

<For client>

- Mandriva Spring 2007 / Microsoft Windows Xp Service Pack 2
- Mozila Firefox 3.0.6 / Internet Explorer 8

Chapter 7. Proposed Work

7.1 Introduction

We propose to develop a web based client server system which can divide a mathematical job into equal or nearly equal parts to generate speed-up in execution time. Thus a heavily loaded mathematical computation can be done distributively in lesser amount of time.

This system is also able to help those naive or sophisticated user who need to perform a heavily loaded mathematical or other type of computation but do not have a cluster to perform them.

This system is quite helpful for those person who have very little idea of C programming language(naive user). It is also able to perform customized jobs for a programmer.

This system document can be used to develop a Beowulf cluster schema if any body want to configure his/her personal cluster.

7.2 Proposed Schema

We have tried to explain our proposed schema with the help of the diagram given below.

In this schema a web server also acting as the master node of the cluster runs a web service which requests a user to input his computation function along with the upper and lower value of data range.

This master node process the data set in nearly equal sub division according to the available resources i.e. the available cluster nodes.

The master node starts execution and also commands the other nodes to start the execution on respective data-set.

Thus we can achieve parallel computation in several nodes of the cluster. This will going to make the execution faster than normal single CPU system.

At the end of execution of all cluster nodes, the result is stored to a

common file. This file is used by master node to gather the result-set and the final output.

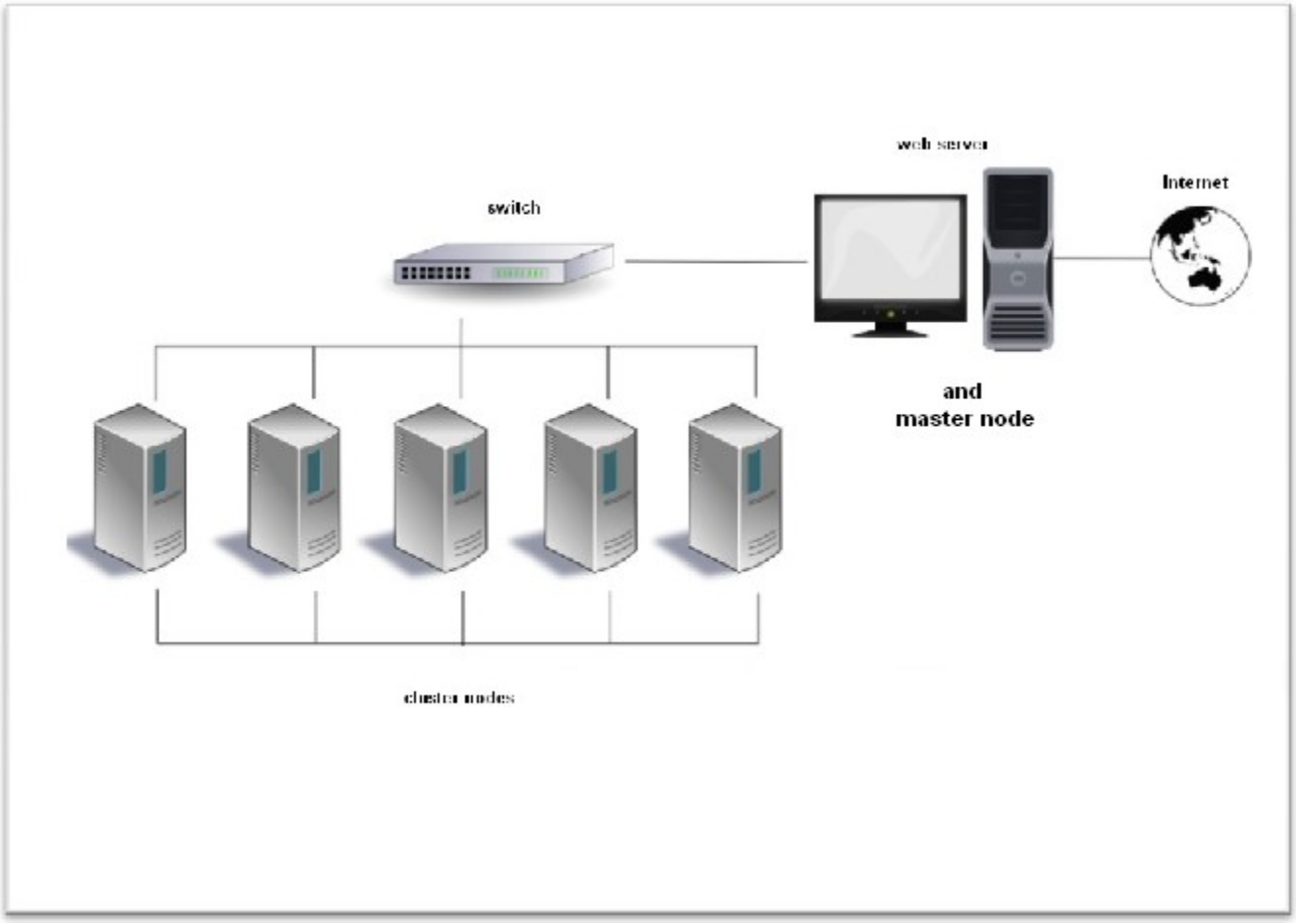


Fig 1

Chapter 8. Proposed Algorithm

8.1 Introduction

In order to achieve the proposed functionality we have proposed some algorithms. Presently there are two algorithm two suffice the functionality. All the algorithm have some predetermined assumptions. They are stated along with the corresponding algorithms.

8.2 First Algorithm

For future references we going to call this algorithm as Data Division Algorithm.

DATA DIVISION ALGORITHM

ASSUMPTIONS

[Every machine has sequential rank (RANK).]

[The size of cluster (SIZE) is known to every machine.]

Begin

1. Calculate D as D: = (RANGE / SIZE).
2. IF RANK = SIZE THEN
a. ISTART := (RANK - 1) * D + 1.
b. IEND := RANGE.
ELSE
a. ISTART := (RANK - 1) * D + 1.
b. IEND := RANK * D.

END

8.3 Second Algorithm

A second algorithm is used in this project to generate the corresponding MPI program from the user program. This algorithm is given below.

ALGORITHM TO GENERATE MPI PROGRAM FROM USER PROGRAM

ASSUMPTIONS

[Every machine has sequential rank (RANK).]

[The size of cluster (SIZE) is known to every machine.]

[An Array of size SIZE is maintained to store result from nodes]

Begin

1.Take input as mathematical expression and corresponding ranges for each variable.

2. Initialize MPI environment <MPI_Init()>.

3. The looping counter ISTART to IEND determined from the dataset division algorithm.

4. Convert the expression to its equivalent C code snippet and execute it in loop from ISTART to IEND.

5. Send the result to store in an common ARRAY, and set its corresponding flag to true, to indicate successful execution.

6. Finalize the program <MPI_Finalize()>

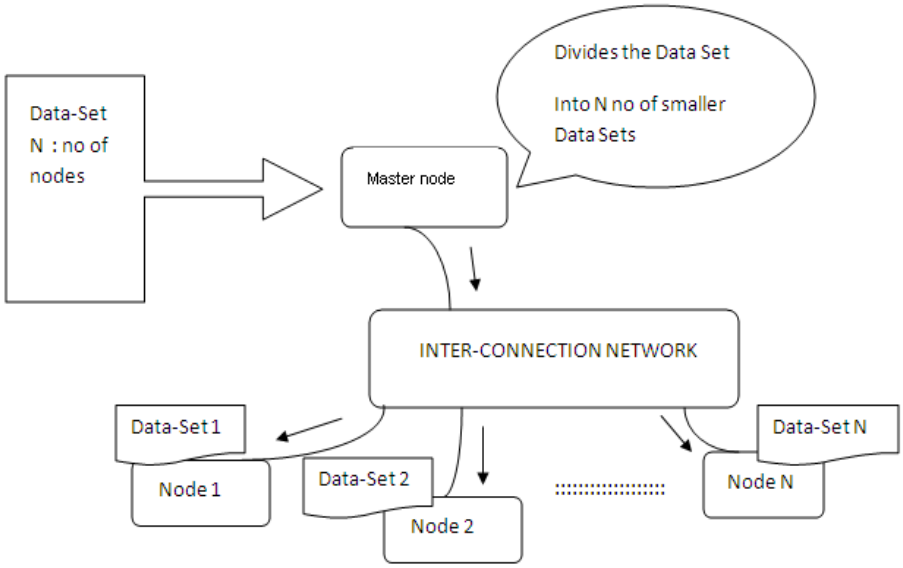
End

Chapter 9. Explanation

9.1 Introduction

In this section of discussion we have explained how our proposed system works.

9.2 Data Division Algorithm Explained



To Show How Data-Set is Divided into Smaller Ones and Distributed among Cluster

fig 2

In our implementation of cluster the user submits his data set [say DS]. DS is huge. So it takes a cumbersome amount of time to completely

execute the entire program.

In this algorithm the entire data-set is divided into smaller sub parts. This algorithm is used for dividing the job into available resources.

This proposal helps us to run the problem in parallel. Thus taking sufficient less amount of time.

But this algorithm is problem specific. For different type of problem must use other methods though the approach will be the same.

9.3 The Proposed System

With the help of above mentioned algorithms we can generate a MPI program.

The generated MPI program/function will be called by each of the cluster nodes. These nodes have their respective data range generated by the Data Division Algorithm. We have allowed only those problems which can be converted into equivalent parallel program. Thus each of the node can successfully execute the job in parallel. Which in term reduces the total working time of process completion.

However our system is not intelligent enough to recognize if the problem can be converted into its equivalent parallel one or not. So we have restricted our system to take input of a particular problem specific way.

Chapter 10. Configuration

10.1 Introduction

One of the major aspect of this project was to configure Beowulf Cluster. In this section we have discussed the nuts and bolts of our cluster configuration.

10.2 Hardware

Hardware choosing is a major task in configuring Cluster. Hardwares should be chosen wisely to get a better performance/cost ratio. If a valuable hardware is chosen which have shown better performance but may have poor performance / cost ratio <PCR>. So it always need a great skill to choose hardwares.

The points one should keep in his/her mind while opting for cluster hardware are given below

- Processor Speed:- Greater processor speed means greater the performance. But to achieve a better performance/cost ratio one can make use of second hand or used processor.
- Network Speed:- As all the processor are connected via network interfaces, So use of faster network technology is always helpful if price is not a constraint for the system. But in order to achieve PCR Ethernet or Gigabyte Ethernet are most suitable choice in compare to optical fibre network or any Wireless network <WiFi>.
- RAM:- As all the nodes of the system are going to be used for heavy computational jobs, they must have sufficient amount of RAM. But the size of RAM is solely dependent on the jobs they are going to perform. Like an image processing job needs a higher amount of RAM in compare to the normal arithmetic jobs.

However in this project we have configured system with available collage resources and our home computers.

10.3 Operating System

Choosing operating system was a relatively easier task. For most clustering job Linux is preferable as it is configurable and free to use. but the dilemma lies in which flavour to be installed. We have chosen Mandriva spring 2007 as it has better GUI and available documentation.

10.4 Configure Node

This is the hard part we faced. Configuring master node can be sub-divide into a number of tasks.

10.4.1LAM/MPI

LAM/MPI is the message passing interface tool we used. So at first this Software is installed in to the system. Installation of this package have a some dependency problem. We have solved this dependency problems.

This tool should be installed in each of the nodes.

10.4.2 SSH

Secure Shell or SSH is the mechanism or software which allows a node to perform job on remote node. Basically the software LAM/MPI uses this tool for its parallel activity.

This software must be configured such it can perform job silently into remote node. which means it will not ask for a password while executing. We have used RSA based authentication for this purpose.

This tool should be installed into each of the cluster nodes. LAM/MPI package must be configured to use SSH. For this a script can added into the user's .bash_profile file.

10.4.3 Name Resolution

It is easier to use name for each machine instead of using IP addresses of the nodes. There are normally two mechanism for achieving name resolution. DNS server and Static naming. For simplicity we have used static naming in our project.

For static naming /etc/hosts file must be altered suitably.

10.4.4 NFS

NFS or Network File System is used to get shared Hard disk space for a network. For successful completion of the task it is necessary to have the program copies in the same path of each node. Besides if each copy of program is different then the output will be erroneous. Thus the program directory is NFS-ed to each of the cluster nodes.

To achieve this master node /etc/exports file must be altered along with all other node's /etc/fstab file.

10.4.5 Apache server

Apache is the free web server which is used in our project to give web-enabled feature to the project.

Our implementation uses PHP for the server side scripting. Thus the Apache configuration file /etc/httpd.conf file must be modified. This software and PHP is only needed to be installed and configured in the master node.

10.4.6 SUDO er

The web-service suffers from some permission problem to read or write file in the server. To get rid of this problem the sudoer file must be changed to give Apache server sudo permission. Actually we have permitted the “apache” user sudo permission.

10.4.7 Service

It is also necessary for the system to have all of these services permitted, like sshd, nfsd, rexec, rlogin. “service” and “chkconfig” commands are used to do this.

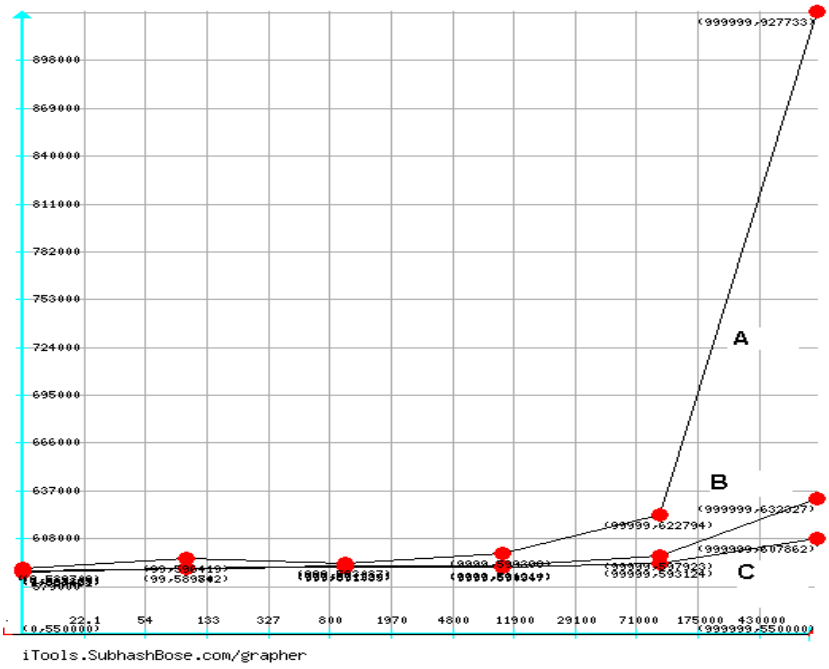
Chapter 11. Experimental Results

11.1 Introduction

In this section we have shown how our proposed system works and the output from the system. To verify our proposals we have build a Beowulf cluster. We also have executed some basic functions with variable range set. The results are given below in a graphical form corresponding to the functions.

11.2 Plot Time v/s Range

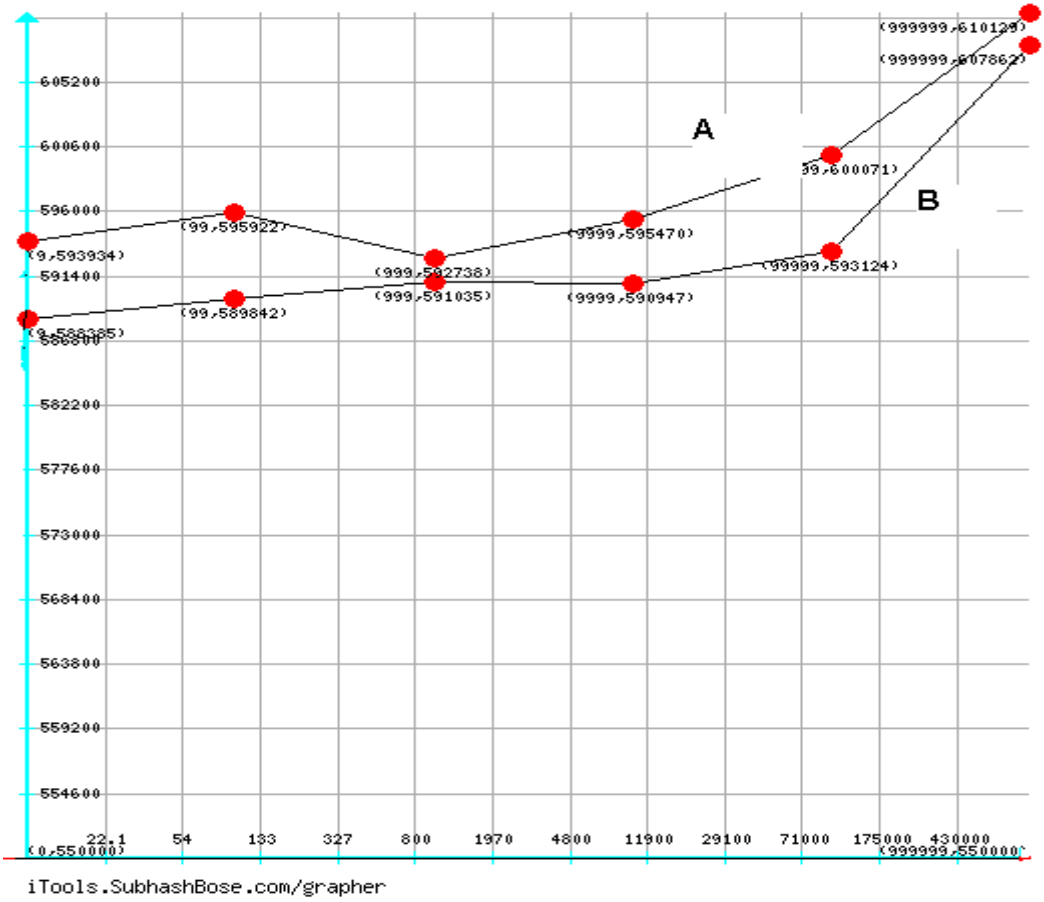
Here is the graph which is plotted Execution time<Y- axis> v/s range<X- axis>. Range is in log scale.



In this graph **A=>Σ(x^x)**
B=>Σ(√x)
C=>Σ(x)

X-axis: Range in log scale
Y-axis: Total execution time(microsecond)
Nature of the curve : Similar in all cases A, B,C . The deviation in their nature is due to execution complexity per cycle

11.3 Plot of Σ(x) for Multiple instance



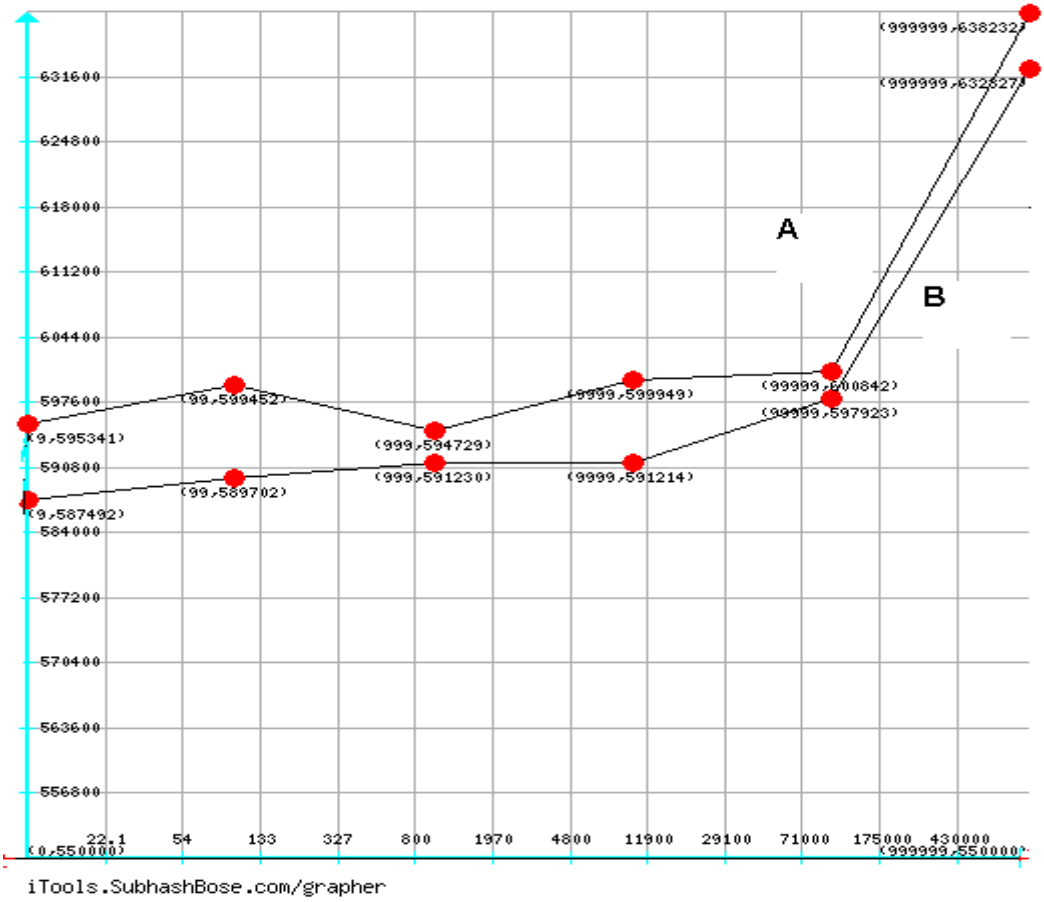
This graph is plotted for the function Σ(x).

A=> With 1 instance of process
B=> With 2 Instance of process

X-axis: Range in log scale
Y-axis: Total execution time(microsecond)
Nature of the curve : Similar in all cases A, B . The deviation in their nature is due to system load.

N.B. System load is very difficult to control. It varies due to the operating system scheduling of that very node's no of active processes.

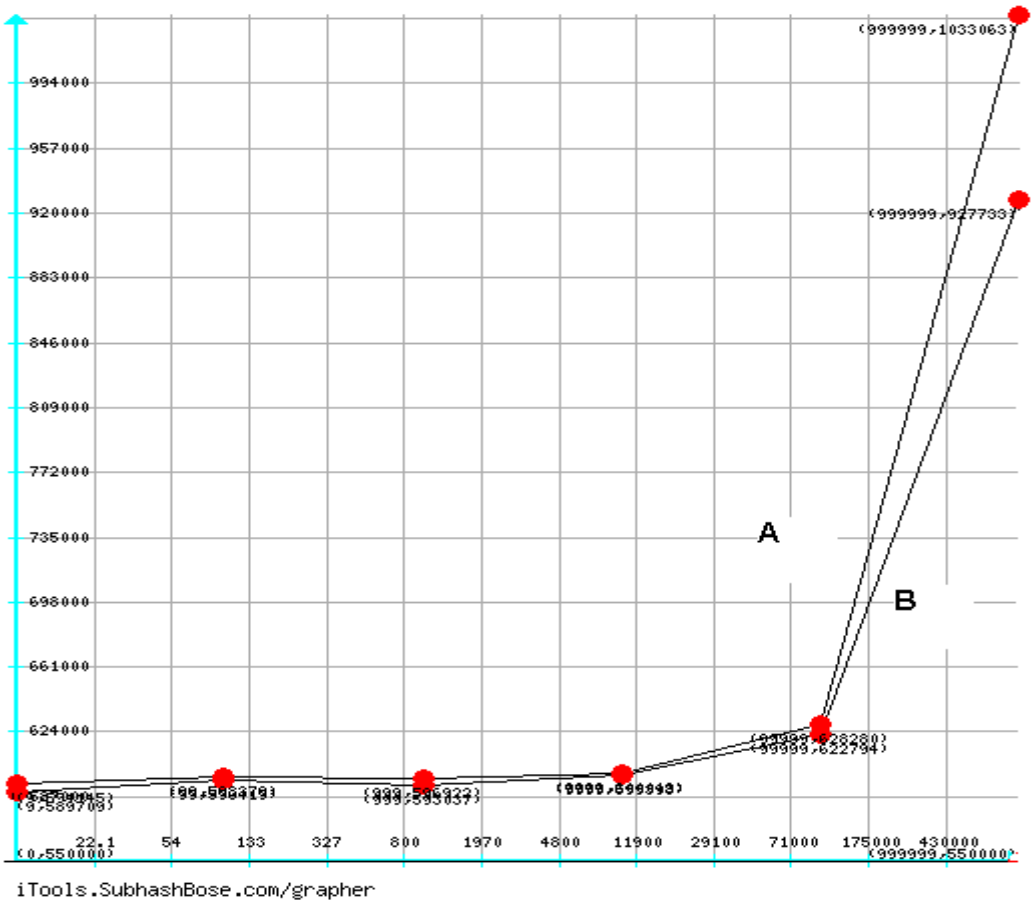
11.4 Plot of Σ(√x) for Multiple instance



This graph is plotted for the function Σ(√x).

A=> With 1 instance of process
B=> With 2 Instance of process
X-axis: Range in log scale
Y-axis: Total execution time(microsecond)
Nature of the curve : Similar in all cases A, B . The deviation in their nature is due to system load.

11.5 Plot of Σ(x^x) for Multiple instance



This graph is plotted for the function Σ(x^x).

A=> With 1 instance of process
B=> With 2 Instance of process
X-axis: Range in log scale
Y-axis: Total execution time(microsecond)
Nature of the curve : Similar in all cases A, B . The deviation in their nature is due to system load.

Chapter 12. Conclusion

12.1 Introduction

Now a days cluster computation is getting more popular day by day. The reason behind this is cluster can give the same throughput of a super computer with much lesser cost. Besides it is scalable. No new kind of hardware infrastructure is necessary. It can be applied with an old computer along with the latest CPU and hardware s. The main advantage is its cost and performance scalability. Our paper is based on the main idea that a simple naïve user can perform his/her huge computation jobs with no a prior knowledge of cluster computation. Basically it achieves the virtualization of a super computer with some cheap custom made hardware

In this project we have configured a system with two heterogeneous hardware. And we have found a satisfactory output from the system. However we have not yet implemented the system in the peer-to-peer model which will be ultimate bench mark to achieve. Besides our implementation presently deals with only static load balancing and static fault tolerance. Hence we can say that our objectives are partially fulfilled.

We are just hoping that our initiative towards cluster computation will be put forward by somebody else. For those who likes this idea we have attached some dimension in which the project can be taken and details of configuring BEOWULF cluster.

Chapter 13. Future Proposal

13.1 Introduction

This portion of our document deals with the proposals which can be achieved in future. It also states the points which can be developed in future. Basically it it is a helpful note for the person who want to work in this field and make a progressive work on this very project.

13.2 Proposals

- Our project deals with the static load-Balancing feature. To make this system for efficient a dynamic load balancing can be achieved. This means load balancing will be achieved on the fly. When a user submits his/her job on this platform the system load will be checked and the job will be distributed according to the load of each cluster node.

- Another load -balancing scheme can can be applied with help of user inference. i.e. An user can select his operating nodes with some provided informatin.

- A dynamic Fault tolerant behaviour can be achieved in this project by using an efficient fault tolerant model or algorithm. This feature will help to recover from any un-wanted situation of fault.

- A peer-to-peer model can be developed such that if a person want to share his/her resource for execution that can be accommodated along with the cluster nodes.

- Presently our implemented system can perform only one task at a time. A further modification can be made to make the system multitasking.

- For any project there is always a scope to modify the user interface. So this system can be made more interactive and user friendly.

Appendix A
Details of Configuration

A.1 Introduction

In the main documentation we have given some theoretical concepts about the configuration of a Beowulf cluster. In this section we are going to explain the basic configuration steps and the required information about them. This is a step by step helping guide to build up a sample Beowulf cluster. The main target of this portion of our work is the software and the file related configuration of a Linux system to set-up the entire environment.

A.2 Create user

We have created a system user “lam” who can perform cluster related jobs. For this a command should be used like “useradd lam ” and then giving password to user lam by ”passwd”.

A.3 Installation of LAM/MPI

LAM/MPI can be downloaded from internet or available with our project file in “lam_home/lam_soft” section. General linux software installation steps are sufficient for this software.

- Unpack the package with “tar -xvf pkgname”
- cd lam7.x.
- “./configure”
- “make”
- “make install”

These steps should be followed for all the necessary tools for this project. Like SSH, APACHE etc.

A.4 Name resolution

For efficient use and for large cluster DNS should be configured. But for small system static name resolution will be sufficient. For this change “/etc/hosts ” in the following format with-out “ s.

	IP	name
e.g.	192.168.0.1	beowulf1

A.5 NFS configure

For configuring NFS “/etc/exports” file should be modified in the following format

DIRECTORY_NAME PERMITED_IP(Permission)
e.g. /home/lam/lam_home *(rw)

For configuring NFS client “/etc/fstab” file of client machine or node machines should be modified in following format with the help of this command

```
cd /etc
cat >> exports
/mnt/wolf 192.168.0.100/192.168.0.255 (rw)
```

A.6 Apache Server configuration

First of all install “PHP 5” and then Apache Http Server. Run this sample script in info.php file and place it “/var/www/html” directory. Isseue the command ”service httpd restart”. Enter “localhost/info.php” into web-browsers url bar. If “PHP” and “Apache” are properly configured the php info should be shown in the web browser.

sample info.php file

```
<?php
phpinfo();
?>
```

All php files should be kept in “var/www/html” location to be hosted.

A.7 Giving SUDO Permission to User

To do this job command will be
”visudo”

and then in the “sudoer” file we have added lines

“apache ALL=(ALL) NOPASSWD: ALL”
“lam ALL=(ALL) NOPASSWD: ALL”

these lines will give user “apache” and “lam” to perform “sudo” command.

A.8 Password less SSH

SSH can be configured in a way that instead of asking for password ,it asks for a pass-phrase. If we use a blank pass-phrase then it is called password less or silent SSH.

To do this we have do as written below

- Use command “skx@lappy:~$ ssh-keygen -t rsa”
- Press enter for each step
  e.g.

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/skx/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/skx/.ssh/id_rsa.
Your public key has been saved in /home/skx/.ssh/id_rsa.pub.
```

- Use command “ssh-copy-id -i ~/.ssh/id_rsa.pub username@machine”.

Appendix B
Introduction to MPI

B.1 Introduction

MPI is complex system with 128 in built function (MPI 1.1). But there are some function or api which are used much more than the others. In this section we will discuss about those important sides of MPI.

B.2 Basic Functions

The basic six function are noted below

- MPI_INIT
- MPI_COMM_SIZE
- MPI_COMM_RANK
- MPI_SEND
- MPI_RECV
- MPI_FINALIZE

A brief description about these functions are given below.

1. int MPI_Init (int *argc, char ***argv)

This function is used to start the parallel execution of LAM universe. Generally it is called in the start of a program.

argc *Pointer to the number of arguments.*
argv *Pointer to the argument vector.*

2. int MPI_Comm_size (MPI_Comm comm, int *size)

This function allows user to get the number of LAM nodes available in the LAM universe.

comm *communicator (handle)*

3. int MPI_Comm_rank (MPI_Comm comm, int *rank)

This function returns the value or rank of the caller node in the entire LAM universe.

comm communicator (handle)

4. **int MPI_Send(void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)**

This is basic send function for MPI environment. Data can be sent between any two nodes via this function.

buf *initial address of send buffer (choice)*
count *number of elements in send buffer (nonnegative integer)*
datatype *datatype of each send buffer element (handle)*
dest *rank of destination (integer)*
tag *message tag (integer)*
comm *communicator (handle)*

5. int MPI_Recv(void *buf, int count, MPI_Datatype datatype, int source, int tag,MPI_Comm comm, MPI_Status *status)

This a also another basic function to receive data from a remote node. MPI_Send and MPI_Recv works in a pair.

6. int MPI_Finalize()

This function is used at the end of MPI program to end the MPI session.

x.3LAM/MPI commands

- “lamboot”:-
  This command is used to activate LAM/MPI modules in the server or client nodes. It is called as starting of MPI environment It requires a boot file which contains the node Ip addresses along with the resolve name.

- “lamclean”: -
  This command is used to clean lam program residue. To recover from previous LAM execution, it is quite helpful.

- “lamgrow”: -
  Used to add a single node to LAM universe.

- “wipe”:-
  Used to clean and shut-down LAM universe. After executing the program this command must be used.

- “mpicc”: -
  This is command to compile a MPI c program.

- “mpiexec”: -
  This command starts a program in LAM environment.

- “mpirun”:-
  This command is the main mechanism to launch MPI processes in parallel.

B.4 LAM/MPI data-type

MPI data-type	C data-type
MPI CHAR MPI	signed char
MPI SHORT	signed short int
MPI INT	signed int
MPI LONG	signed long int
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	

Some More MPI Functions

1. MPI_Abort: Terminates MPI execution environment.

   Terminates all MPI processes associated with the communicator comm; in most systems (all to date), terminates all processes.

2. MPI_Address: Gets the address of a location in memory. This routine is provided for both the Fortran and C programmers. On many systems, the address returned by this routine will be the same as produced by the C & operator, but this is not required in C and may not be true of systems with word- rather than byte-oriented instructions or systems with segmented address spaces.

3. MPI_Comm_create: Creates a new communicator.

4. MPI_Comm_get_name: return the print name from the communicator.

5. MPI_Comm_group: Accesses the group associated with given communicator

6. MPI_Comm_remote_size: Determines the size of the remote group associated with an inter-communictor.

7. MPI_Comm_set_name: Give a print name to the communicator.

8. MPI_Comm_spilt: Creates new communicators based on colors and keys.

9. MPI_File_close: Closes a file

10. MPI_File_delete_: Deletes a file.

11. MPI_File_iread: Non-blocking read using individual file pointer.

12. MPI_File_iwrite: Non-blocking write using individual file pointer.

13. MPI_File_open: Opens a file.

14. MPI_File_read: Read using individual file pointer.

15. MPI_File_sync: Causes all previous writes to be transferred to the storage device.

16. MPI_File_write: Write using individual file pointer.

17. MPI_Finalize: Terminates MPI execution environment.

18. MPI_Init: Initialize the MPI execution environment.

19. MPI_Sendrecv: Sends and receives a message.

20. MPI_Start: Initiates a communication with a persistent request handle .

21. MPI_Startall: Starts a collection of requests.

22. MPI_Test: Tests for the completion of a send or receive.

23. MPi_Testall: Tests for the completion of all previously initiated communications.

24. MPI_Wait: Waits for an MPI send or receive to complete.

25. MPI_Waitall: Waits for all given communications to complete.

26. MPI_Waitsome: Waits for some given communications to complete.

Reference

1. https://computing.llnl.gov/tutorials/mpi/

2. http://www.wikipedia.com

3. http://www.beowulf.org

4. http://www.dhpc.adelaide.edu.au/projects/beowulf/

5. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC523878/

6. http://doi.ieeecomputersociety.org/10.1109/EMPDP.2003.1183627

7. http://www.gridbus.org/~alchemi

8. http://archive.ncsa.illinois.edu/Cyberia/MetaComp/MetaHome.htm

9. *http://www.mcs.anl.gov/research/projects/mpi/www/www3/Web pages for MPI Routines.htm*

10. *http://www.lam-mpi.org/tutorials/nd/part1/part1.pdf*

11. *http://www.lam-mpi.org/tutorials/nd/part2/part2.pdf*

12. http://www.lam-mpi.org/tutorials/nd/part3/part3.pdf

13. http://www.sapac.edu.au/systems/overview.html

14. http://www.nd.edu/~parasite/tcp.pdf

15. http://tldp.org/HOWTO/html_single/Beowulf-HOWTO/

16. http://www.debian-administration.org/articles/152