Design and implementation of a distributed computing infrastructure using commodity hardware components and Open Source Software within the University of Las Palmas de Gran Canaria

Fidel García[1] Enrique Rubio[1] Antonio Ocón[1] Luis Álvarez[1] and Manuel J. Galán[1]

Abstract

In this paper we make a study of the computer devices known as "clusters". We focus on a special cluster initially developed by T. Sterling and D. Becker named "Beowulf Cluster". Next we enumerate the several advantages of such device. Finally we make a proposal about the implementation of a Beowulf Cluster using commodity equipment dedicated to run high-performance computing tasks at the University of Las Palmas de Gran Canaria.

Introduction

Concurrency and Parallelism

Regarding program execution, there is a very important distinction that needs to be made: the difference between "concurrency" and "parallelism". We will define these two concepts as follows:

- "Concurrency": refers to the parts of a program that can be computed independently.
- "Parallelism": the parallel parts of a program are those "concurrency" parts that are executed on separate processing elements at the same time.

The distinction is very important, because "concurrency" is a property of the program and efficient "parallelism" is a property of the machine.

Ideally, "parallel" execution should result in faster performance. The limiting factor in parallel performance is the communication speed (bandwidth) and latency between computing nodes.

Many of the common parallel benchmarks are highly parallel thus communication and latency are not the bottle neck. This type of problem can be called "obviously parallel". Other applications are not so simple and executing "concurrent" parts of the program in "parallel" may actually cause the program to run slower, thus offsetting any performance gains in other "concurrent" parts of the program. In simple terms, the cost of communication time must pay for the savings in computation time, otherwise the "parallel" execution of the "concurrent" part is inefficient.

Now, the task of the programmer is to determining what "concurrent" parts of the program should be executed in "parallel" and what parts should not. The answer to this will determine the efficiency of the application.

In a perfect parallel computer, the ratio of communication/processing would be equal to one and anything that is "concurrent" could be implemented in "parallel". Unfortunately, real parallel computers, including shared memory machines, do not behave "this well".

Architectures for Parallel Computing

The following description of parallel computing architectures is by no means exhaustive, we are only giving the basic definitions of some concepts that will be used later.

HARDWARE ARCHITECTURES

There are three common hardware architectures for parallel computing:

- Shared memory machines, SMP, that communicate through memory, i.e. MPP (Massively Parallel Processors, like the nCube (see [1]), Convex SPP (see [2]), Cray T3D, Cray T3E (see [3]), etc.). This kind of configuration is sustained on dedicated hardware. The main characteristic is a very high bandwidth between CPUs and memory.
- Local memory machines that communicate by messages, i.e. NOWs (Networks of Workstations) and clusters. In this category each workstation maintains its individuality, however there is a tight integration with the rest of the members of the cluster. So we can say they constitute a new entity known as "The Cluster". In our proposal we will focus on a particular kind of cluster called "Beowulf Cluster" (see [4]).
- Local memory machines that integrate in a loosely knit collaborative network. In this category we can include several collaborative Internet efforts which are able to share the load of a difficult and hard to solve problem among a large number of computers executing an "ad hoc" client program. We can mention the SETI@home (see [5]), Entropia Project (see [6]), etc.

The former classification is not strict, in the sense that it is possible to connect many shared memory machines to create a "hybrid" shared memory machine. These hybrid machines "look" like a single large SMP machine to the user and are often called NUMA (Non Uniform Memory Access). It is also possible to connect SMP machines as local memory compute nodes. The user cannot (at this point) assign a specific task to a specific SMP processor. The user can, however, start two independent processes or a threaded processes and expect to see a performance increase over a single CPU system. Lastly we could add several of this hybrid system into some collaborative Internet effort building a super hybrid system.

SOFTWARE ARCHITECTURES

In this part we will consider both the "basement" software (API) and the application issues.

Software API (Application Programming Interface)

There are basically two ways to "express" concurrency in a program, i.e. Messages and Threads:

A Message is a simple entity: some data and a destination processor. Common message passing APIs are PVM (see [7]) or MPI (see [8]). Messages require copying data while Threads use data in place. The latency and speed at which messages can be copied are the limiting factor with message passing models. The advantage to using messages on an SMP machine, as opposed to Threads, is that if you decided to use clusters in the future it is easier to add machines or scale up your application.

Threads were developed because shared memory SMP designs allowed very fast shared memory communication and synchronization between concurrent parts of a program. In contrast to messages, a large amount of copying can be eliminated with threads. The most common API for threads is the POSIX API. It is difficult to extend threads beyond one SMP machine, It requires NUMA technology that is difficult to implement.

Other methods do exist, but the former are the most widely used. It is important to remember that the expression of concurrency is not necessarily controlled by the underlying hardware.

Application Issues

In order to run an application in parallel on multiple CPUs, it must be explicitly broken into concurrent parts. There are some tools and compilers that can break up programs, but parallelizing codes is not a "plug and play" operation. Depending on the application, parallelizing code can be easy, extremely difficult, or in some cases impossible due to algorithm dependencies.

Definition of Cluster

A previously stated a cluster is a collection of machines connected using a network in such a way that they behave like a single computer. Clusters are used for parallel processing, for load balancing and for fault tolerance. Clustering is a popular strategy for implementing parallel processing applications because it enables companies to leverage the investment already made in PCs and workstations. In addition, it is relatively easy to add new CPUs simply by adding a new PC or workstation to the network.

The Beowulf Cluster

Beowulf is not a special software package, new network topology or the latest Linux kernel hack. It is a kind of cluster built primarily out of commodity hardware components, running an OSS (Open Source Software) (see [9]) operating system like Linux or FreeBSD, interconnected by a private high-speed network, dedicated to running high-performance computing tasks.

One of the main differences between a Beowulf Cluster and a COW (Cluster of Workstations) is the fact that Beowulf behaves more like a single machine rather than many workstations. The nodes in the cluster don't sit on people's desks; they are dedicated to running cluster jobs. It is usually connected to the outside world through only a single node.

While most distributed computing systems provide general purpose multi-user environments, the Beowulf distributed computing system is specifically designed for single user workloads typical of high end scientific workstation environments.

Beowulf systems have been constructed from a variety of parts. For the sake of performance some non-commodity components (i.e. produced by a single manufacturer) have been employed. In order to account for the different types of systems and to make discussions about machines a bit easier, It has been proposed the following classification scheme:

CLASS I: This class of machines are built entirely from commodity "off-the-shelf" parts. We shall use the "Computer Shopper" (see [10]) certification test to define commodity "off-the-shelf" parts. (Computer Shopper is a 1 inch thick monthly magazine/catalog of PC systems and components). A CLASS I Beowulf is a machine that can be assembled from parts found in at least 3 nationally/globally circulated advertising catalogs.

The advantages of a CLASS I system are:

- Hardware is available form multiple sources (low prices, easy maintenance).
- No reliance on a single hardware vendor.
- Driver support from O.S. commodity usually based on standards (SCSI, Ethernet, etc.).

The disadvantages of a CLASS I system are:

- Best performance may require CLASS II hardware.

CLASS II: This class is simply any machine that does not pass the Computer Shopper certification test.

The advantages of a CLASS II system are:

- Performance can be quite good

The disadvantages of a CLASS II system are:

- 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. In the last times we are seeing an increment in the number CLASS II Beowulf Clusters using 64-bit ALPHA Processors due to the large performance increase that can be achieved.

Evolution of the Beowulf Cluster: State of the Art

In the summer of 1994 T. Sterling and D. Becker, working at CESDIS (Center of Excellence in Space Data and Information Sciences) (see [11]) under the sponsorship of the ESS (Earth and Space Sciences) project (see [12]), built a cluster computer consisting of 16 DX4 processors connected by channel-bonded Ethernet. They called their machine Beowulf. The machine was an instant success and their idea of providing COTS (Commodity Off The Shelf) base systems to satisfy specific computational requirements quickly spread through NASA and into the academic and research communities. The development effort for this first machine quickly grew into a what we now call the Beowulf Project. Some of the major accomplishment of the Beowulf Project will be chronicled below, but a non-technical measure of success is the observation that researchers within the High Performance Computer community are now referring to such machines as "Beowulf Class Cluster Computers". The Beowulf Project is now hosted by Scyld Computing Corporation, which was founded by members of the original Beowulf team with a mission to develop and support Beowulf systems in the larger commercial arena.

The first Beowulf-class computers that achieved the gigaflops goal appeared at Supercomputing '96 in Pittsburgh. One of those came from a collaboration between Caltech and the Jet Propulsion Laboratory and the other from Los Alamos National Laboratory. Both systems consisted of 16 200-megahertz Pentium Pro processors and were built for about $50,000 in the fall of 1996. One year later, the same machines could be built for about $30,000.

In a paper for the 1997 supercomputing meeting – simply called SC97 – Michael Warren of Los Alamos and his colleagues wrote: "We have no particular desire to build and maintain our own computer hardware. If we could buy a better system for the money, we would be using it instead."(see [13]).

Finally we can mention that the Avalon, which is a co-operative venture of the Los Alamos National Laboratory Center for Nonlinear Studies and Theoretical Division, built as a 140 64-bit processors Alpha Beowulf Cluster machine appears as the 265th in the list of the fastest computer systems in the word (see [14]).

There are a lot of Beowulf Clusters spread around the word, dedicated to every kind of computationally intensive task. Among them we can mention:

- Stone SouperComputer Oak Ridge National Lab (ORNL) a 126 node cluster at zero dollars per node. The system has already been used to develop software for large-scale landscape analysis (see [15]).
- The SuperAbacus: an implementation in the CityU Image Processing Lab at City University of Hong Kong. To support multimedia signal processing (see [16]).
- LoBoS Supercomputer for Molecular Graphics and Simulation Laboratory, National Institutes of Health NIH, (see [17]). This cluster is dedicated to study more complex biological systems using computational methods.

Beowulf Clusters are also deployed in our country (Spain), they are also used for intensive computation. The most mature projects could be:

- HIDRA University of Barcelona's UB-UPC Dynamical Systems Group dedicated to several projects that require a huge amount of computations (i.e., numerical simulations of continuous and discrete systems, bifurcation analysis, numeric and symbolic computation of invariant manifolds, etc.) (see [18]).
- LAMA's Materials Laboratory at UPV/EHU running Monte Carlo simulations of phase transitions in condensed matter physics (see [19]).

Characteristics of the Beowulf Cluster

Commodity networking, especially Fast Ethernet, has made it possible to design distributed-memory systems with relatively high bandwidths and tolerably low latencies at low cost.

Free operating systems, such as Linux, are available, reliable, and well-supported, and are distributed with complete source code, encouraging the development of additional tools including low-level drivers, parallel file systems, and communication libraries.

With the power and low prices of today's off-the-shelf PCs and the availability of 100/1.000 Mb/s Ethernet interconnect, it makes sense to combine them to build High-Performance-Computing and Parallel Computing environment.

With free versions of Linux and public domain software packages, no commercially available parallel computing system can compete with the price of the Beowulf system.

The drawback to this system is, of course, that there will not exist any "support center" to call when a problem arises (anyway, "support centers" are many times only marketing hype and do not provide real support). We can say that the Open Source Support Center is the whole Internet, in the sense that there does exist a wealth of good information available through FTP sites, web sites and newsgroups. Besides that you can also sign a maintenance agreement with any of the increasing number of companies that provide commercial support to these installations.

Another key component contributing to forward compatibility is the system software used on Beowulf. With the maturity and robustness of Linux, GNU software and the "standardization" of message passing via PVM and MPI, programmers now have a guarantee that the programs they write will run on future Beowulf Clusters, regardless of who makes the processors or the networks.

That said the main characteristics a Beowulf Cluster can be summarized in the following points:

- Very high performance-price ratio.
- Easy scalability.
- Recycling possibilities of the hardware components.
- Guarantee of usability / upgradeability in the future.

Our proposal of Beowulf Cluster for the University of Las Palmas de Gran Canaria.

All the good characteristics of the Beowulf Cluster justify its deployment in any organisation that require high computational power. In the case of an academic institution we can say that it is not only advisable but imperative. The Beowulf cluster is not only a wonderful tool to provide high computing power to the University but, at the same time, is a very interesting objects of study "per se". The evaluation of its performance, adaptability, scalability, its behaviour regarding the parallelization of procedures, etc. Is a field of study that we suspect full of findings.

In our case we propose the initial deployment of a first step Beowulf Cluster made up of a small number of hardware components that can be eventually incremented.

System Description

Our system has a hardware part and a software part. Hardware part consists in eight PCs connected by a Fast Ethernet switch at 100 Mb/s. One of this PCs is the cluster's console that controls the whole cluster and is the gateway to the outside world (master node). Nodes are configured and controlled by the master node, and do only what they are told to do.

The proposed node configuration consists of an AMD single processor based PC at 750 Mhz, with 256 megabytes of RAM and including a local IDE hard disk drive of 8GB capacity. They have also a non expensive video card and floppy disk drive. Besides they will be provided with a Fast Ethernet network interface card.

The master node is provided with a larger hard disk drive (24 MB) and a better graphics video card, besides it has a second network interface and also CD-ROM drive, medium sized monitor and keyboard and mouse to be able to perform the controlling tasks for the cluster.

The proposed Fast Ethernet switch will have 24 autosensing ports and will be SNMP capable.

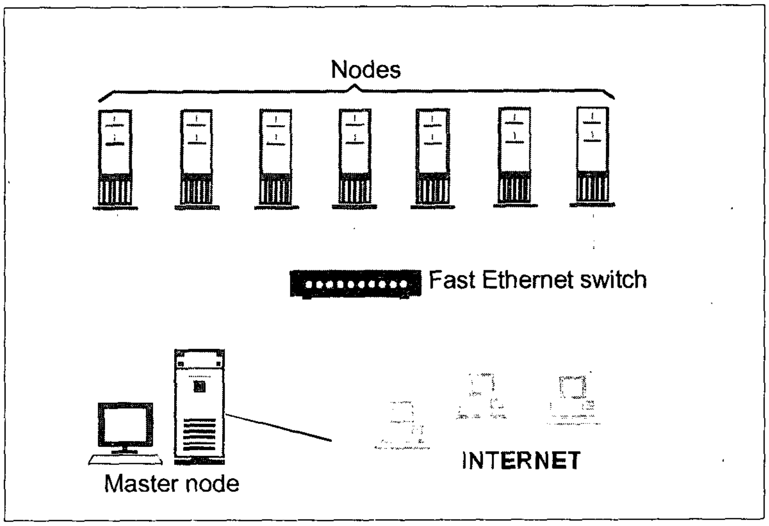

Fig. 1. Beowulf Cluster Hardware Components

In the description of the hardware we can see that only one node needs input/output devices. The second network interface card in the master node is used to connect the Intranet to the Internet. The switch has a number of port bigger than strictly necessary to can enlarge the cluster in the future.

The logical part will be built using GNU / Linux operating system according to the distribution "Extreme Linux CD" (see [20]) with additional OSS software such as kernel modifications:

- PVM: Parallel Virtual Machine: PVM is a software package that permits a heterogeneous collection of Unix / Linux or NT computers hooked together by a network to be used as a single large parallel computer. It is freely-available, portable, message-passing library generally implemented on top of sockets. It is clearly established as the de-facto standard for message-passing cluster parallel computing (see [7]).
- MPI libraries: Message Passing Interface: Communication between processors on a Beowulf Cluster is achieved through the Message Passing Interface (MPI). This is a standardized set of library routines. Both the C and the Fortran programming languages are supported (see [8]).

Additionally we will be proceed to the installation of several configuration, management and monitoring tools which make the Beowulf architecture faster, easier to configure, and much more usable.

Basic Software Installation and Configuration

As previously stated the installations pathway will run along the "Extreme Linux CD". The following steps will be taken:
The first step we will installing the Master Server which involves the following tasks:

- Partition sizes.
- Installing Red Hat Linux.
- Network configuration.

    - Setting up DNS.
    - Network file configuration: "/etc/hosts", "/etc/resolv.conf" and "/etc/hosts.equiv".
- Local file configuration: ".cshrc".
- Clock synchronization.

After the installation of the master server we will proceed to the installation and configuration of the client nodes:

- Installing the operating system on one client.
- Cloning clients.
- Configuring clients.

The third step will be installation of basic application software:

- Compilers.
- Communication Software: PVM and MPI.
- Conversion Software.
- System Monitoring Software: bWatch, httpd and CGI scripts, Netpipe, netperf, NASA parallel Benchmarks, CMS.

Finally we will attend the security concerns both in the master sever and client nodes.

Fields of Application

There are a lot of OSS software packages optimized to run in clusters like Beowulf. We can mention the following:

- *MP_SOLVE, Parallel Sparse Irregular System Solvers* solving large, irregular, sparse, indefinite systems of equations with multiple excitation vectors on distributed memory parallel computers using LU factorization (see [21]).
- *NAMD* is a parallel, object-oriented molecular dynamics code designed for high-performance simulation of large biomolecular systems (see [22]).
- *POV-RAY, The Persistence of Vision Raytracer* is a high-quality tool for creating stunning three-dimensional graphics (see [23]).
- *FFTW* is a C subroutine library for computing the Discrete Fourier Transform (DFT) in one or more dimensions, of both real and complex data, and of arbitrary input size (see [24]).

Nevertheless there are many other applications that can take profit when run in a Beowulf Cluster, their range covers from standard numerical applications, going through high intensive physical and chemical computation, biochemical modeling and multimedia and CAD applications.

Conclusions

In the present paper we have made a quick and succinct overview about the state of distributed computing, centering our focus on a concrete cluster configuration called "Beowulf Cluster".

The advantages of this class of cluster configuration are evident for any organization that requires high computational power "for the buck". This is, when we take into account the performance/price ratio, easy scalability and upgradeability and recycling properties of the hardware components. If this is true for any organization, we are convinced that it is imperative for an academic institution like our University. Therefore we make a proposal of deployment of such a device starting with a schematic installation to be eventually enlarged and improved.

References

[1] nCUBE, http://www.ncube.com/
[2] Convex, http://www.convex.com/
[3] Cray Inc., http://www.cray.com/
[4] The Beowulf Project, http://www.beowulf.org/
[5] Search for Extraterrestrial Intelligence (SETI), http://www.kevlar.karoo.net/seti.html.
[6] Entropia.com Inc., http://www.entropia.com/
[7] Parallel Virtual Machine (PVM), http://www.epm.ornl.gov/pvm/pvm_home.html
[8] LAM / MPI Parallel Computing, http://www.mpi.nd.edu/lam/
[9] N. Drakos, "*Debunking Open-Source Myths: Development and Support*", http://gartner3.gartnerweb.com/public/static/hotc/hc00088469.html, 2000.
[10] Computer Shopper.com, http://www.zdnet.com/computershopper/
[11] Center of Excellence in Space Data and Information Sciences, http://cesdis.gsfc.nasa.gov/
[12] Earth and Space Sciences project (ESS), http://sdcd.gsfc.nasa.gov/ESS/overview.html
[13] Michael Warren, "*Pentium Pro Inside: I. A Treecode at 430 Gigaflops on ASCI Red, II. Price/Performance of $50/Mflop on Loki and Hyglac*" , SC97.
[14] TOP500 Supercomputer Sites, http://www.netlib.org/benchmark/top500/top500.list.html, 1999.
[15] Stone SouperComputer Oak Ridge National Lab (ORNL), http://stonesoup.esd.ornl.gov/
[16] SuperAbacus, http://abacus.ee.cityu.edu.hk/
[17] LoBoS Supercomputer, National Institutes of Health NIH, http://www.lobos.nih.gov/
[18] HIDRA University of Barcelona's UB-UPC Dynamical Systems Group, http://www.maia.ub.es/dsg/hidra/index.html
[19] LAMA's Materials Laboratory at UPV/EHU, http://lcdx00.wm.lc.ehu.es/~svet/beowulf/
[20] Extreme Linux CD, ftp://beowulf.gsfc.nasa.gov/mirror/extreme_linux/
[21] MP_SOLVE, Parallel Sparse Irregular System Solvers, http://emlab2.nmsu.edu/mp_solve/
[22] NAMD, http://www.ks.uiuc.edu/Research/namd/
[23] POV-RAY, The Persistence of Vision Raytracer, http://www.povray.org/
[24] FFTW, http://www.fftw.org/
[25] Jacek Radajewski and Douglas Eadline, "*Beowulf HOWTO*", http://www.linux.org/help/ldp/howto/Beowulf-HOWTO.html, 1998.
[26] Jacek Radajewski and Douglas Eadline, "*Beowulf Installation and Administration HOWTO*", http://www.beowulf-underground.org/doc_project/BIAA-HOWTO/Beowulf-Installation-and-Administration-HOWTO.html, 1999.

1. CICEI. Centro de Innovación en Tecnologías de la Información . University of Las Palmas de Gran Canaria. Edificio de Ingenierías, Campus Universitario de Tafira Baja. 35017 Las Palmas de G.C. SPAIN. (www.cicei.ulpgc.es), (fidel@polaris.ulpgc.es).