High performance clustering with Linux

David Santo Orcero
irbis@df.ibilce.unesp.br
irbis@activanet.es

November 23, 2000

Abstract

On our talk we are going to review the posibilities of Linux as a tool for high performance clustering. We are going to analize the availability of compilers, applications, libraries, and finnaly we are going to review some parallel supercomputer facilities on Linux, as Linux Beowulf Clusters and Linux Mosix Clusters.

Contents

1 Overview

2 Preamble

2.1 Disclaimer

THIS DOCUMENT IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIMERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVELIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACONNECTION WITH THE DOCUMENT OR THE USE OR OTHER DEALINGS WITH THE DOCUMENT. IT MIGHT BE COMPLETLY WRONG, OR IT WOULD BURN YOUR HARD DISK. YOU WILL USE ITS INFORMATION AT YOUR OWN RISK.

2.2 Copyright

This licence is free -from freedom of speech-, but this is not a GPL licence. If you want to do something special with the document that is not on this licence, please ask me. Sure that you will can do, as far as you keep a reference of the authory of the document.

LICENCE TERMS OF THIS DOCUMENT

You are allowed to use this document at your own risk, on the following terms:

1. . **You are freely allowed to use this document for your own, but if you use this document to do scientific research and you publish something, you must cite this document on your paper.** The full citation is marked bellow.
2. You are freely allowed to improve this document, and distributed the improvement, as far as you do ALL the following things:
   (a) You must keep this licence on the destination document. Anyway, the part that you improve or append can include further restrictions based on more works that must to be cited, but the new whole document must remain free.
   (b) You must indicate clearly what parts of the document you improve, and what parts of the document you include. It must be clear what parts of the document are original.
   (c) You must keep special care on do not modifiing neither the autory, neither this licence text. It must be clear my authory

of the main document, and that any derivated work must cite the work that I indicated on this HOWTO.

(d) Derivated works can not be distributed on a printed media -book, journal- without my written permision. The permision will be usually granted, however, in the same conditions that printed versions descrived below.

3. You are freely allowed to distribute this document in any electronic form, as far as you do ALL the following things:

   (a) You must keep it unmodified. If you modify it, you fall on the part of the licence described avove.

   (b) If you do a product -Linux distribution, or same thing similar- and you sell it, I must recive a certification of the inclusion of my work -for curriculae- and a copy of the final version without any cost of my own on the address indicated below -but please ask to me before because I can move to other house-. For non-profit uses you can ask to me to find another agreement on this point.

4. **You are freely allowed to print this document to your personal use.** Distribution of the printed form falls on the term descrived bellow.

5. You are freely allowed to include my work on books, use this work as part of a paper, and publicate this work in any print media as far as you do ALL the following things:

   (a) You keep it unmodified, and you print the full text, specially the part of the authory and the licence. If you want to print part of the document or the document improved, you must ask to me first.

   (b) If you are going to print and distribute this document on a print form, and this document is more or equal of the 80% of the work, I must appear as the only author of the final printed work.

   (c) If you are going to print and distribute this document on a print form, and this document is less of the 80% of the work, I must appear as author of the final printed work. I must be the first author of the final printed work if the % of this document on the final printed work is equal or greater than 1/numberOfAuthors *100 of the final printed work.

   (d) In any of the cases before, I must recive a certification of the inclusion of my work -for curriculae- and a copy of the final version without any cost of my own on the address indicated below -but please ask to me before because I can move to other house-. For non-profit uses you can ask to me to find another agreement on this point.

The work that you must cite for your publications is at:
http://www.biocristalografia.df.ibilce.unesp.br/irbis
Or ask to me at mailto:irbis@activanet.es or mailto:irbis@df.ibilce.unesp.br
The address to send the certifications and the copies is:
David Santo Orcero
Departamento de Física
IBILCE/UNESP
Rua Cristovião Colombo, 2265
São Jose de Rio Preto, SP
15054-000
Brazil

I am only a student with a fellowship. If you cite my work, the chance of maintain my fellowship becomes greater. Keeping my fellowship will allow me to continue writting free documentation and free software for you. Thanks.

2.3 About the author

David Santo Orcero was born in 1974, but he was reborn in 1986 when he discovered a ZX-Spectrum. His father said to him that he ought to programme the computer if he wanted to play with it. One month later, his father said that he ought to play with the computer if he wanted to programme. He spend his first year learning BASIC. After his first year, he began to code on Z80 assembler. Assembly and disassembly at hand was not for being cool, but for he didn't find a ZX80 assembler. The Sinclair QL saw his firsts programmes being terminated -early contact with "Software engineering crisis"-. After that, he used an Commodore Amiga , and his first PC computer with MS-DOS and after this, OS/2. He began to use Prolog at 1989, and it became his first programming language during six years. Finally, he converted himself to the true religion, and he began to learn C. Meanwhile, in 1994, he did his first Linux install. He had used SLS, Slackware, RedHat, Debian, Conectiva and Mandrake. Finally, he got his Ms. Sc. degree on Computer Science at Málaga University (Spain), and travel to Brazil, where he is living now. Meanwhile, he has been Unix/Linux teacher in FORMAN (Spain) and in UFG (Brazil), and he has administrated the most weird combination of Unix boxes in some scientific laboratories, always being a strong advocate of free software philosophy.

Nowadays, when he isn't being bitten by mosquitoes, he does his Dr. Sc. degree on molecular biophysics on IBILCE/UNESP (Brazil).

He has a FAPESP fellowship in the research project “Computational methods for molecular biophysics”. He research developing a killer-app that would be delivered as free software as far as they were finished -no later than 2002, or my advisor will (scientifically) kill me-. That killer-app will allow to calculate protein structures and protein functionality using information collected by SAXS and X-ray scattering techniques, with an intuitive and easy-to-use interface. He also created and maintains a Mosix cluster that is being used by his research team to study snakes toxines.

2.4 Acknowledges

“Es de bien nacidos ser agradecidos” - Spanish proberb

There are some persons and entities that I must to cite here. Maybe you will be here on the next version...

- My wife Maria Francisca Ribeiro de Araujo Santo Orcero, for her understanding and support.
- My father Antonio and my mother Maria del Carmen, for all.
- FAPESP, for giving to me a grant for my Ph.Dr., on the project “Computational methods for molecular biophysics”. Using the *reserva técnica* of this grant -arround R$4000- I built the first Mosix cluster.

2.5 Is this talk useful for me?

In computing science, the most of the problems can be organizated on families, with similar solution. This talk is useful for you if you:

- Develop basic science using computers for this.
- You are involved on “number-crushing” or “big-challenge” problems, you need more power and you have no more money.
- You find interesting to gain knowledge on an exiting area, that is going to be one of the more important engines of the development of humankind on XXI century.

2.6 Overview of the talk

A overview of we are going to talk on the first half is:

- Supercomputers facilities for cheap cost. We are going to compare some Linux clustering techniques, their strong and weak points, and which one can be best for a particular problem.

- Some about what I really do. Maybe the speaker is doing something interesting. ;-)
- If we have time -I do not thing so- some neat hints about how to administrate clusters for high-performance or high-availability where you are the only one computer science guy. Dilbert rulez. ;-)

Finally, If I do not talk something, it would be because short of time or short of knowledge. Do not doubt to share knowledge on the Q&A session at the end of my presentation. Sharing knowledge is the basis of science and free software movement.

On the second half we are going to do a live demostration of the instalation and configuration of a Mosix cluster.

2.7 Some basic concepts

2.7.1 The free software paradigm, GNU and FSF

- After some troubles with his institute's software development policy, MIT hacker Richard Stallman founded FSF –Free Software Foundation–.
- He created the GPL licence and the copyleft. The sense of our "free" is about freedom of speech, not Free beer.
- Free software movement claims that development of software must be free. You must be allowed to improve the code of the others, as you keep the register of their work.
- Free software movement looks as basic science development paradigm. No one can patent Newton's Law. You *discover* software –what is about knowledge–, not *make* software –what is about industrial machinery–.
- Free Software way of thinking is common on Computational Chemistry, only look at CCP collaborative projects of UK government, CCL, QCPE –born on 50's, before Stallman's FSF–, GAMESS, MOPAC...
- You use lots of FSF tools; GNU C compiler, Emacs...

2.7.2 The Bazaar Model

- It is the theoretical basis that try to explain the fast growing of some pieces of Free Software –Apache Web Server, Linux Kernel–.
- Proposed by Eric Raymond, an open source evangelist.

- Development of traditional Software is like building a Cathedral. Its management is centralized, as the masons only build whatever they were said to be built.
- The new model: Bazaar model. Decentralized management, lots of people doing its own.
- It has historical cases on Computational Chemistry: GAMESS, CCP4. And It worked!
- Bazaar development has one problem: nobody does what nobody want to do –eg, there are lots of GUIs; but FSF GPLed kernel, Hurd, ten years after the work begun, is far to be completed–.
- Eric Raymond's model has flaws: it does not explain Motzilla's disaster.
- One of its stronger points is the mass of parallel beta-testers with knowledge and ability of correct and improve the source code.

2.8 In the begining, there was free software...

The free software, and the concept of sharing it is not strange to science.

Sharing it is not a strange concept on science. Sharing knowlede is the backbone of the advance of science and quality of life of humankind.

- Do you imagine a patent over the weel?
- Do you imagine a patent over the writting?
- Do you imagine a patent over the potery?
- Do you imagine a patent over the gravitation law?

Patents, and any limitations to the expansion of knowledge, used to be strange to basic science.

And software was always considered by scientists as a piece of knowledge.

2.9 The fever of Linux's fashion, and some Linux Miths.

One day, by a mistery reason, Linux was no more a extravaganza and became cool. But we must renember that:

- **Linux is a flavour of Unix**: With its strongness and its weakness.

- **Managing Linux needs deep knowledge**: As you have more power on your hands, you must to learn how to manage it. The good tools of administration, as *Linuxconf*, make lots easier to manage Linux networks for the expert, but, for example, a person that knows nothing about DNS services, can't configure a DNS server for a domain.
- **Using Linux needs some knowledge**: No way. As in the case avove, as you have more power, you must know what you are doing.

All this affirmations are not "cool" for non-technical users. And most of them are mixed with Linux miths. Anyway, nobody makes MS-Windows better than M$, thus if we begin to modify all the things that make Linux powerfull, changing for Windoze things, we will have more Windows again.

Personally, I have heared for a important and famous person of Linux orbit "X-Window sucks". He also have said "Linux sucks" and other things like this, thus it looks that all that it is not his product sucks. Anyway, it is a strong stream of opinion on some circles that Linux should look and act like Windows, but being free. It is the "grandma is at the keyboard" stream.

By other way, the power and the flexibility of X-Protocol and Unix capacities will save the day on the day-to-day use on a laboratory. We change some time of learning for better features.

On the other side, important distributions, with M$-Like instalation procedures, needs more time and knowledge to be instaled than other more **user friendly** -IMHO, only more pretty- distributions.

Many times you are forced to install one operating system or distribution acording to your boss's opinion about what is more user friendly. No way that most of them looks the same after installing. It doesn't matter.

Maybe, with all this fever of Linux's fashion, people that really loves Linux and Free Software continues being misfits. But nowadays only these misfits can take the best of the hardware.

2.10 The right to read

When our freedom was in danger, Stallman did an excelent article called "the right to read". This article, far from loosing actuality, each day is more important, because we are arriving to the situation descrived on it. When I read that italian society of authors wanted to cut the free libraries, I renembered Stallman's article and began to feel afraid.

2.10.1 The 4 freedoms

The objetive of Stallman since then was to create a free operating system, based on his believes on Freedom. He said that all software must to have the follow freedoms:

- **Freedom 0**: The freedom to run the program, for any purpose. (It is somewhat stupid, but nowadays it must being specificated because this freedom is in trouble, as all the others)
- **Freedom 1**: The freedom to study how the program works, and adapt it to your needs. Access to the source code is a precondition for this.
- **Freedom 2**: The freedom to redistribute copies so you can help your neighbour.
- **Freedom 3**: The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. Access to the source code is a precondition for this.

3 Linux as a supercomputing tool

3.1 Free operating systems that can be used as supercomputing tool

There are many free Unix flavours. They have diferent free licences; but the more common licences are BSD and GPL, completly free. There are slight differences between them on the performance; but they are very similar for a scientist. The more importante diference is on the instalation process, because Linux has the easiest instalation process. The most common free operating systems that can be used for cheap supercomputing are:

- *Linux*: Under GPL licence, it is the most common free operating system available, and the easier one to install. It has lots of different distributions, each one with its strongness and weakness. It is the best choose for a desk scientific workstation on a cheap computer. More information can be found at *http://www.linux.org/*.
- *Hurd*: It is the same system than Linux, but it uses the Hurd kernel, developed by FSF; Linux uses the kernel developed by Linus Torwalds. The Hurd is built on top of CMU's Mach 3.0 kernel and uses Mach's virtual memory management and message-passing facilities. It has a really creative design, so creative that at this time it does not work well. Anyway, It may be an alternative on the future. Linux binaries are not compatible with Hurd

ones, but most of them can be recompiled for Hurd. Optimal for research on Computer Science, other uses are discouraged at this moment. It is GPLed. It can be found at *http://www.gnu.org*.

- *FreeBSD*: Free flavour of Unix, under BSD licence. It has a faster implementation of TCP/IP stack -with a great diference-. Anyway, it is harder to install. Nearly all the aplications for Linux can be recompiled for FreeBSD. Maybe is the best choose for servers -like SAMBA, NFS or Web servers-. It can be found at *http://www.freebsd.org/*.
- *NetBSD*: Free flavour of Unix, under BSD licence. Really mature and stable. It is the free flavour of Unix more portable, thus more ported to more platforms. Currently, it has versions on alpha amiga, arc, arm26, arm32, atari, bebox, cobalt, hp300, hpcmips, i386, luna68k, mac68k, macppc, mipsco, mvme68k, news68k, newsmips, next68k, ofppc, pc532, pmax, prep, sgimips, sh3, sparc, sparc64, sun3, vax, x68k. The best if we want easily port Unix to a strange machine. It can be found at *http://www.netbsd.org/*
- *OpenBSD*: Free flavour of Unix, under BSD licence. It is, by distance, the most secure operating system that can be found. It is the only operating system which source code has being fully audited. It had three years without a remote hole in the default install, and only one localhost hole in two years in the default install. It supports binary emulation of most programs from SVR4 (Solaris), FreeBSD, Linux, BSD/OS, SunOS and HP-UX. It is the best for machines that can be attacked, as mail servers, DNS servers, NIS servers, firewalls... More information at *http://www.openbsd.org/*.

3.2 Strong points of Linux as supercomputing tool

Nowadays, Linux is the best operating system to do supercomputing, high-availability uses or high-performance uses with low budget.

- **Existence of free Fortran 77 compilers.** The most of the scientific libraries are in Fortran, and the most of the scientific programs are also in Fortran, due to historical reasons. Most of the computational chemistry programmes are written in Fortran, and nobody at this moment is thinking about porting the gigantic QCPE dinosaurs-packages to C.
- **Existence of free C compilers.** C is fast and cool. And today there are lots of people developing code using C.

- **Stability.** A real calculation on computational chemistry problem need lots of time to be completed. Some problems, like folding problems, are especially CPU-intensive. If the machine hangs all day, you can not use the machine for serious research. Linux is really stable, and that thing allows you to do long simulations on your PC desktop.
- **Performance.** Many workstations are known for the low user time response. A cheap PC may need 40 minutes of CPU, but you wait only 45 minutes. A AIX workstation used by M.D. students, Ph.D. students, postdocs, researchers, will need 15 minutes of CPU, but you will wait a couple of hours. Final performance is better.
- **Availability.** All of us have a PC on our desktop. Few of us has an parallel supercomputer on our desktop. Anyway, if you want the supercomputer and you have the money, you should wait long customs delays and taxes. You can buy just-in-time a PC on any store, and maybe your work will delay more, but it will be finished before the supercomputer is here.
- **Availability of PVM.** There exists a complete and efficient implementation of PVM library. PVM is free and well-documented, and it is available as Debian, RPM and tallball, with lots of free documentation and its wrappers to Fortran. Some Linux distributions include PVM –SuSE, RedHat PowerTools, Extreme Linux project–. Usually it is easier to recompile a parallel package for a pile of cheap Pcs with Linux without native support and documentation, than recompiling the same package for a supercomputer.
- **Availability of MPI.** As on the case of PVM, there exists lots of applications that make use of MPI and that can be directly recompiled under Linux. MPI's situation is completely different than PVM's one. On PVM we had only one full implementation. On MPI, we have some different implementations. All of them use to work fine under simple applications; but some applications can need features for which a determined implementation lacks of support. The four most common free implementations of MPI standards are MPITCH, LAM, CHIMP and UNIFY. Unify is somewhat special. It is a PVM wrapper for MPI calls, that allows as emulate a MPI as mix PVM and MPI calls on the same code.
- **Supercomputing facilities.** Finally, Linux have some supercomputing facilities. The most important but less appreciated is that Linux support most of Intel-based multiprocessor motherboards as part of the stable line of Kernels. But this facility is

not the only one. We have two cheap-supercomputer projects, Beowulf project –from CESDIS, NASA– and Mosix project –from the Hebrew University of Jerusalem–. The two ones use completely different points of view.

- **Optimal MIPS/$.** The most important one. Ask for the price of a Linux box. Ask for the price of a supercomputer. Look for the cost for MIP. Only as a example, we can see the table that come with this article. Other example: The last 9 of March of 1999, on LinuxExpo 1999, Beowulf class supercomputers outperformed to Cray supercomputer. The mundial record of PovRay –an important ray tracing application, that involves heavy floating point calculations– benchmark, hold by a Cray processing T3t-900-AC64 of $5.5 million, was broken by a Beowulf class supercomputer of $150,000. The Beowulf cluster rendered the image three times faster that the Cray supercomputer. There are hundreds of other examples.

There exists other advantages, but form part of the advantages of Linux and Free software. Price of hardware, spare parts and reparations, on-line support, global quality... Sure you know lots of them.

3.3 Problems with Linux as a supercomputing tool

Anyway, there exists limitations of Linux as a supercomputing tool. The most important ones are:

- **Lack of any free Fortran parallelizing compiler.**
- **There does not exist a free Fortran-90 compiler.** Fortran 90 is not as popular as its antecesor, but it is broadly used on scientific media.
- **Weakness of MPI**. At this time, there exists some problems with free implementations of MPI, that lacks of support of threaded MPI extensions. This is a serious problem, because several important computational chemistry parallel packages uses threaded MPI as programme API.
- **Weak knowledge about what means Linux and free software.** It is true that Linux has been strongly supported by community, but at this time there are lots of people that do not want hear about Linux, and consider Linux "something bizarre of the computer guy".

4 Number-crushing and supercomputing with Linux

Linux is used usually on machines of low power. This does not means that Linux can be used in an Alpha processor, but that it is not the more frecuent scenario. The most frecuent scenario is having no enought budget.

Using Alpha processors have also some other problems. Some countries, as Brazil, have a draconian politics of importation. Importation and customs' steps may delay obtain the machine months, sometimes nearly a year.

Customs taxes also uses to be extremily high, and somewhat absurd. As an example, **I was taxed 5 months ago for 400US$ for bringing to Brazil my 4-years-old pentium!**. The worst is that the same brazilian law assures that I hadn't to pay nothing, but you are phoned and they say to you that you will lost your computer if you don't pay in a week, and you finish paying to recover your HD and its information.

There are some other dark histories on customs, like trying to get a camera for forgeiner tourists -particulary, my father's video camera-, and so on...

Bring the hardware using a foundation avoid you to pay taxes, but you need nearly a year between you pay and you get the machine.

Then, sometimes it is not only prize. It is **availability**. You can not be one year stopped, because finacial organizations that granted to you don't what to know why you spend a year without publishing.

There are other countries, like Cuba or China, where some supercomputer facilities have limitations for exportation from the US.

Linux, as it is available on high-availability machines, it is perfect for that kind of countries. Neither the most close goverment can block science with importation politics, neither any country can block for you to buy a machine to do science because you are on his list of "non-fair-with-me" countries; because you can always go to the computer shop of the next door, buy a pile of PCs and begin to do supercomputers, without need neither greats amounths of money nor ask permision to the local goverment or to US goverment.

And the best: you don't need to pay the "M$ tax".

A Linux box has a reasonably use of the calculus power of a single machine. M$ uses to say that it is not true, but it is best to try NT and Linux on the same machine that earing M$ FUD techniques.

By other way, you can't reach the performance of an Alpha with one single pentium. Sorry, but it is the reality. Linux is not Lourdes neither Fatima sanctuary. Let's see how can we get more power.

4.1 SMP support

SMP support of Linux have a very bad press. I really do not know why, because I work dayly with that kind of machines, and they ARE stable if the memory is good. And if the memory is bad they are unestable with all OS, because it is a hardware problem.

It is true that Linux do not support the must out of the line hardware SMP facilities. It is also true that it has support only for NT (neither 95, nor 98).**It is also true that I had never known anybody with such hardware.** If I could spend so much money, I would buy an Alpha processor, and stop playing with Intel toys.

It is also true that SMP support is Beta. But "Beta" is a relativistic term. Masquerading has been beta code for years, and I have been used dayly masquerading for years without any trouble or bug. Other operating sistems have release code for its internal functions, and hangs twice per day, on the best of the situations, in a production environment.

SMP is a simple way to do parallelism on Linux. But it is far away to be the cheaper. Let's see others.

4.2 Clustering

Really, it is difficult to find one kind of aplication that could not take advantage of using clustering tecnologies. From high-end computer -that needs clustering for performance- to the low end user -that would be beneficied on data recovery- all users can be beneficiated of a inteligent cluster aplication.

Let's see what kind of clusters we can find, and for what is useful each one.

4.3 Beowulf, doing supercomputing with a pile of PCs

Beowuf is an architectural model of cheap supercomputing, based on GNU/Linux and other free software comodities. It was proposed by researchers of the Center of Excellence in Space Data. Originaly, it only needs common hardware, but it works better with non-ethernet network solutions, because with an ethernet network cann't scale more that 16 processors.

4.3.1 Performance's analysis of Beowulf class supercomputers

Comparative of NAMD performance

Processors	1	2	4	8	16	32	64	128	160	192
Cray T3E										
Time	x	6.12	3.10	1.60	0.810	0.397	0.212	0.123	0.098	x
Speedup	x	(1.97)	3.89	7.54	14.9	30.3	56.8	97.9	123	x
SGI O2000										
Time	8.28	4.20	x	x	0.542	0.271	0.152	x	x	x
Speedup	1.0	1.96	x	x	15.3	30.5	54.3	x	x	x
ASCI-RED										
Time	28.0	13.9	7.24	3.76	1.91	1.01	0.50	0.279	0.227	0.196
Speedup	1.0	2.01	3.87	7.45	14.7	27.9	56.0	100	123	143
HP735/125 (ATM)										
Time	24.1	12.4	6.39	3.69	x	x	x	x	x	x
Speedup	1.0	1.94	3.77	6.54	x	x	x	x	x	x
Cluster Linux PII (fast Ethernet)										
Time	12.34	6.42	3.28	1.69	1.01	0.62	x	x	x	x
Speedup	1.0	1.92	3.77	7.30	12.21	19.90	x	x	x	x

Execution time in seconds for a system of 36573 atoms, with a cut-off of 12A.

Conclusions:

- **Scalability of results.** Fairly good, but could be best using a Gb net, better than Fast Ethernet.
- **Comparation between the result of PII table and above table.** It looks that you need the double of processors to reach the same performance, than a O2000, and the same processors to reach a T3E.
- SGI O2000/Linux ratio; Linux Cluster with the double of processors.

  Time: for 1 procs in O2000: 1.28; for 2 procs in O2000: 1.28; for 16 procs in O2000: 0.87.

  Excellent! Beat O2000 with se same prize!
- Cray T3E/Linux ratio; Linux Cluster with the **same number** of processors

  Time: for 2 procs in T3E : 0.95; for 4 procs in T3E : 0.94; for 8 procs in T3E : 0.94; for 16 procs in T3E : 0.8; for 32 procs in T3E : 0.68

  SIMPLY MARVELOUS!!!

4.3.2 The great flaw of the Beowulf arquitecture

The Beowulf architecture, in the case of an optimal load balancing between the different process of the different nodes, assures the optimal performance of the application at low cost. But, unfortunately, this case is not always the best possible stage. The most common stage is that some nodes has CPU-bounded jobs, some nodes has IO-bounded nodes, and that is in that way because the born and the finish of process. That means that some nodes can be subused meanwhile other nodes might be overloaded.

The scenario can be worse. In the case of nodes of different performance, the most powerful nodes uses to be misused on normal task partitions. And in the case of multiuser environments, the problem is that we have no control about the load of the different nodes.

The best would be having the best of Beowulf clusters, but with load balancing. That is where Mosix clusters plays a great role.

4.4 The Mosix architecture

The Mosix architecture was developed by prof. Ammon Barak team on Hebrew University of Jerusalem, and it gives to Linux clusters the better of clustering. The features of Mosix are:

- **Automatic load balancing of process**. Mosix asures the best use of all the resources of the cluster;
- **Memory ushering**. Process will run wherever they have space, for not to swap;
- **Descentralized control** There is not a central authority;
- **Common interprocess space**. A Mosix cluster looks like an alone machine to the user;
- **Backwards compatibility**. It can be used PVM, MPI and sockets; all the clustered and Beowulf applications runs on a Mosix without modifying code;
- **Fork and forget programming model**. Creating a Mosix parallel aplication is so easy as calling the *fork()* Unix POSIX function;
- **Transparent to the user**; and, finnaly, the applications can give clues of where is their bottlenecks, for Mosix optimize them; but this is not compulsatory.

4.4.1 Performance of the Mosix architecture

Two comparations -figures at the end of the article-; PVM vs. Mosix, a test battery of different applications with different runtimes, that proves that a Mosix outperforms PVM on a cluster; and running a molecular dynamics simulation of 1000000 atoms, on a Pentium Pro cluster with Ethernet, that proves that Mosix outperforms a SP/2. For 8 node, Mosix is 4 times faster than SP/2. When we use 16 nodes, we find again a bottleneck on the network layer, that can be easly broken using Fast Ethernet -A 30R$ communications card!!-. Mosix is a fully-competitive platform for CPU-bonded parallel tasks.

4.5 Network interfaces for Linux clusters

When we are going to build a dedicated cluster, it is not a good idea to use a common Ethernet adaptor. Neverthless it is cheap and reliable, big bandwidth applications will find a bottleneck here when we have 16 nodes or more. There are some good network interfaces, more expensive than Ethernet, but they increase dramatically the overall performance of our cluster, and that's why they are strongly recommended.

Today, best solution is Myricom net, but tomorrow may vary. Anyway, we must to renember that the PC bus clock may be a bottleneck, limiting the speed to 133Mbps –33MHz busspeed, today nonwritten standard–. IBM NetInfinity 7000 server series can run four times faster, arise to a 64 bits, 66MHz PCI bus, but not all cards are able to afford this.

4.5.1 Myricom

- Myricom cards and switches that interconnect at speeds of up to 1.28 Gbps in each direction.
- It has two different forms: copper-based and optical.
- The copper version for LANs can communicate at full speed at a distance of 10 feet but can operate at half that speed at distances of up to 60 feet.
- The fiber version can operate at full speed up to 6.25 miles on single-mode fiber, or about 340 feet on multimode fiber.
- It has some different network configurations: direct point to point, hub-based, or switch-based.
- Its average latency between two directly connected nodes is 5 to 18 microseconds, a magnitude or more faster than Ethernet.
- Its switches can be connected in cascade, and this only increments the latency between segments.

4.5.2 Giganet

- Giganet network was promoted by Intel.

- It uses its own network protocol, discarding IP, but this is transparent to the software.
- It is not WAN routable.
- It is able to do 1 Gbps unidirectional communications between the nodes at minimum latencies of 7 microseconds.

4.5.3 IEEE SCI

- IEEE SCI has a ring-topology-based networking system unlike the star topology of Ethernet.
- It is faster to large-scale communication.
- It can be used with a torus-2D or a torus-3D topology.
- It is the best choose for high number of nodes, because it can support hundreds or thousands of nodes.
- It has low latencies (under 2.5 microseconds), and it can run at 400 MB per second (3.2 Gbps) in each direction.

4.6 APIs for Beowulf and Mosix clustering programming

Nearly all parallel APIs and communications APIs can be used on Linux. Anyway, the most popular are the Sockets interface, PVM and MPI. The three APIs can be used as on Beowulf clusters as on Mosix clusters.

Really, on Mosix clusters, we can use nearly all communications API, due to the "fork and forget" programming model. If we do a heavy use of "/proc/mosix" filesystem, we can take the best of our Mosix clusters. Anyway, at this time it is used a more general approachment: using standard parallel APIs to alow compatibility with Beowulfs and COWs –Clusters of Workstations–.

4.6.1 Sockets

The sockets are the faster API. But, unfortunately it does not exist a sockets port to Fortran. It is fully portable between Unix platforms; but it has incompatibilities between Unix sockets and Windows sockets and problems with data encoding between different architectures. Anyway, pairs **sprintf** and **sscanf** can save the day, sending in ASCII 7 bits, because ASCII is now a standard.

4.6.2 PVM — Parallel Virtual Machine

PVM has an excellent Linux port, and is fully standard. It is also the non-official parallel programming standard. It can be used in the most of the languages, including Fortran. Maybe PVM is somewhat slow, but it is easy to learn and use. All programs that use PVM are directly executables in parallel fashion in Beowulf and Mosix clusters.

4.6.3 MPI — Message Passing Interface

MPI is the official industry standard. Unfortunately, there is not a full free implementation of the standard for Linux. There are some partial implementations for Linux, each one with its strong points and weakness. If you find a MPI lib that works fine with your application, OK. If you don't, you are in a problem. MPI has an specially problematic threaded MPI support. At the time this paper was written, it had no free support.

4.6.4 Fork

Mosix allows to avoid headaches, as one of its stronger points. It is the so-called: "fork-and-foget" model. Basically: Create new tasks with fork(), communicate them by files on a common share filesystem and enjoy! This only works on Beowulf clusters, not on Mosix clusters.

4.7 Linux Virtual Server

The Linux Virtual Server project has implemented a number of kernel patches that create a network load-balancing system for incoming TCP/IP traffic. The Linux Virtual Server software examines incoming traffic, and based upon a load-balancing algorithm, redirects that to a set of servers acting as a cluster. That allows network applications such as Web servers to run on a cluster of nodes to support a greater number of users.

On a certain way, it is completly complementary to the Mosix model. Meanwhile Mosix migration is based on process migrations, and sockets can not migrate, on a Linux Virtual Server only can migrate the socket.

It is important that all Linux Virtual Server must be attached at the same segments. If we want to use more complex topologies, as multilevel topologies or hive topologies, we will need tunneling IP packets. But sometimes we will need that kind of topologies due to the scalability limit of Ethernet networks, thus it can be a non-trivial issue making a medium-sized Linux Virtual Server cluster with Linux Virtual Server.

4.8 TurboLinux, TurboCluster and enFuzion

TurboLinux has a product called TurboCluster that was originally based on the kernel patches developed by the Linux Virtual Server project. That means that it allows most of the same benefits, and it has the same drawbacks as the originating project. But TurboLinux has also developed some tools for monitoring the behavior of the cluster help to manage the cluster. It also has commertial support, that Linux Virtual Server hasn't.

EnFuzion is an upcoming scientific clustering product from TurboLinux that isn't based on Beowulf. It is suposed that it does support hundreds of nodes, runs any existing software, and it doesn't need custom parallel applications written to the environment. It supports automated load balancing and resource sharing between the nodes and failed jobs can be automatically rescheduled.

4.9 Linux-HA Project

The objetive of Linux-HA project is not related with neither time of response nor velocity. They are completly different of the two before. The key is: whatever happends, keep the system running. They are used when is critical that the machine runs, and the machine must resist to hardware and software fall-offs. It is the general case of servers, but more specificaly of web servers, internet service providers, and in science on computers that control experimental machineries.

The project has software that can maintain heartbeats between nodes and take over IP addresses of failed nodes. Should a node fail, it uses the Fake Redundant IP software package to add the address of the failed node onto a working node to assume its responsibilities. Thus the failed node can be replaced automatically in a matter of milliseconds. For practical use, that heartbeat is usually kept in the several seconds range, unless you have a dedicated network link between the nodes. At that point, the user applications on the failed system still need to be restarted on the new node. It is also full supported by a firm.

4.10 Some stravaganza

Here we can find some of the weird APIs and products that can be found on the market. At this time there not exist any real aplication over them, but some day maybe we can find an aplication that uses some of them. Or maybe cann't.

4.10.1 Java

There exists some Java material, very interesting for theoretical computing science, but not very usefull for number crushing problems. Maybe next century...

4.10.2 Cplant

A new proposal of Sandia labs. Uses new protocols for nearly all, to break the limits of TCP/IP and related protocols. Only for Alpha, only using portals, but can break the TeraFLOP barrier. At his moment, it is not a broadly extended project due to the cost of the cluster and that there is not free aplications for it. Old aplications do not work over Cplant.

4.10.3 Legion

Metamodel of paralelism. At this time, there is not any aplication that uses it. Legion is an attempt to build a true multicomputer system. That is a cluster whereby each node is an independent system, but the overall system appears to the user as a single computer, as Mosix inteeds; but Legion was designed to support a single worldwide computer consisting of many hosts and many software objects. Within Legion, users can construct their own collaborative groups. On its theoretical side, Legion provides high-performance parallelism, load-balancing, distributed data management, and fault-tolerance. It supports high-availability through its fault-tolerance management and dynamic reconfiguration across the member nodes. It also has an extensible core that can be dynamically replaced or upgraded over time as new advancements or developments arise. The system is not under a single monolithic control but can be managed by any number of organizations, each supporting an autonomous part of the whole. The Legion API provides high-performance computing through its built-in parallelism.

All this stuff rely over the user's computer operating system and negotiates between the local and the distributed resources. It handles resource scheduling and security automatically and also manages a context space to describe and access any object out of all of the possibilities in the entire system. However, it does not need to run under system administrator privileges on each node and can work within the confines of a nonprivileged user account. That increases the flexibility of the nodes and users that can join Legion.

This is what Legion team says. At this time, there is not any application, neither real nor toy, over legion.

4.10.4 PAPERS

As part of a project of MIT, they try to show us that the most of the cost of communication was sincronization, not sending information. PAPERS was build over parallel port interface to show this. It looks that works, but I haven't seen any real aplication over PAPERS.

4.10.5 Portals

A interesting proponsal, strongly defended by SANDIA people. It tries to break TCP/IP and PVM limitations on scalability. At this time, it works in some interesting proyects at SANDIA. Neither of them are Free Software, then we will have to wait and see. It should be peer-reviewd before we could evaluate it.

4.10.6 Unify

Wrappers of MPI over PVM. The power of this stuff is that we can use MPI and PVM and the same time. It is somewhat old, and it is being superated by another proponsals before it had been used on a real aplication.

4.11 Caveats about Linux clustering...

...or when is better to buy an Alpha.

We will have a lot of work in the case of MPI for some aplications, because it lack of full implementation of the standard. The solution, unfortunately, is to by a propietary implementation MPI that implements all the protocol (there is one).

Anyway, we would do better buying a huge Alpha SMP or an O2000 than a Linux cluster if:

- We have the money.
- We have a serial -non parallel- aplication.
- None of the parallel aplications equivalent to our serial aplication is good for us. (strange, but it can happend).
- The time of parallelizing and executing our aplication will be greater than importation of an Alpha and execution.