Clustering

Remo Suppi Boldrito

PID_00148472

Universitat Oberta de Catalunya
www.uoc.edu

Index

Introduction

A computer cluster refers to a group of computers working closely together with a common aim. These computers consist of hardware, communication networks and software for working together as though they were all part of one single system. There are various reasons for which it would be desirable to set up these clusters, but one of the main ones is so as to be able to process information more efficiently and quicker, as though it were a single system. Generally, a cluster works on a local area network (LAN) and provides efficient communication, although the machines will be located close to each other physically. A bigger version of the concept is the grid, where the aim is the same, but it involves groups of computers connected to each other through a wide area network (WAN). Some programmers think of the grid as a cluster of clusters in a 'global' sense. Although the increasingly advanced technology and decreasing costs make it easier to set up these types of systems, the complexity and efforts required to use dozens or hundreds (or, in some cases, thousands) of computers are very great. However, the advantages in computing time mean that, despite this situation, these types of high performance computing (HPC) solutions are considered very attractive and are constantly developing. In this unit, we will show some of the most widely spread and used approaches. [Rad, Dieb, Prob, Prod, Proe, Gloa]

Note

A cluster is a group of computers working closely together, often connected on a LAN.

Grids are groups of computers connected with wide area networks (WAN).

1. Introduction to High Performance Computing (HPC)

The advances in technology have resulted in fast, low-cost and highly efficient processors and networks, which have brought about a change in the cost/performance ratio in favour of using interconnected processing systems in a single high-speed processor. This type of architecture can be classified into two basic configurations:

- *Tighly coupled systems*: these are systems in which the memory is shared by all the processors (shared memory systems) and the memory of each processor is 'seen' (by the programmer) as one single memory.

- *Loosely coupled systems*: they do not share memory (each processor has its own) and they communicate through messages passed through a network (message passing systems).

The first example is known as a parallel processing system and the second as a distributed computing system. In the latter case, we can say that a distributed system is a set of processors interconnected on a network in which each processor has its own resources (memory and peripherals) and they communicate by exchanging messages on the network.

Computing systems are a relatively recent phenomenon (we could say that computing history started in the seventies). Initially, they consisted of large, heavy, expensive systems, which could only be used by a few experts and they were inaccessible and slow. In the seventies, advances in technology led to some substantial improvements carried out using interactive jobs, time sharing and terminals and the sizes of the computers were reduced considerably. The eighties were characterised by a significant improvement in the performance and efficiency (which has continued to today) and a dramatic reduction in the sizes, with the creation of microcomputers. Computing continued to develop through workstations and advances in networking (from 10 Mbits/s LANs and 56 Kbytes/s WANs in 1973 to today's 1Gbit/s LANs and WANs with asynchronous transfer mode (ATM) and 1.2 Gbits/s), which is a fundamental factor in current multimedia applications and those that will be developed in the near future. Distributed systems, for their part, originated in the seventies (systems with 4 or 8 computers), but really became widespread in the nineties.

Although administrating/installing/maintaining distributed systems is a complicated task, given that they continue to grow, the basic reasons for their popularity are the increase in performance and efficiency that they provide in inherently distributed applications (due to their nature), the information that

can be shared by a group of users, the sharing of resources, the high fault tolerance and the possibility of ongoing expansion (the ability to add more nodes to gradually and continuously increase the performance and efficiency).

In the following sections we will look at some of the most common parallel/distributed processing systems, as well as the programming models used to generate code that can use these features.

1.1. Beowulf

Beowulf [Rad, Beo] is a multi-computer architecture that can be used for parallel/distributed applications (APD). The system basically consists of a server and one or more clients connected (generally) through Ethernet, without using any specific hardware. To explore this processing capacity, it is necessary for the programmers to have a distributed programming model that, whilst it is true that it is possible to do this through UNIX (socket, rpc), may require a very significant effort, given that the programming models are at the level of systems calls and C language, for example; but this working method can be considered as low-level.

The software layer provided by systems such as parallel virtual machine (PVM) and message passing interface (MPI) facilitates significantly the abstraction of the system and makes it possible to program parallel/distributed applications easily and simply. The basic working form is master-workers, in which there is a server that distributes the task that the workers perform. In large systems (systems with 1,024 nodes), there is more than one master and nodes dedicated to special tasks such as, for example, in/out or monitoring.

Note

Various options:
- Beowulf
- OpenMosix
- Grid (Globus)

One of the main differences between Beowulf and a cluster of workstations (COW) is that Beowulf is 'seen' as a single machine in which the nodes are accessed remotely, as they do not have a terminal (or a keyboard), whereas a COW is a group of computers that can be used by both the COW users and other users interactively through the screen and keyboard. We must remember that Beowulf is not software that transforms the user's code into distributed code or that affects the kernel of the operating system (like Mosix, for example). It is simply a way of creating a cluster of machines that execute GNU/Linux and act as a supercomputer. Obviously, there are many tools that make it possible to achieve an easier configuration, library or modification to the kernel for obtaining better performance levels, but it is also possible to build a Beowulf cluster from a GNU/Linux standard and conventional software. The construction of a Beowulf cluster with two nodes, for example, can be achieved simply with the two machines connected through Ethernet using a hub, a standard GNU/ Linux distribution (Debian) and the network file system (NFS) and after enabling the network services such as rsh or ssh. In such a situation, we might argue that we have a simple two node cluster.

1.1.1. How do we configure the nodes?

First, we must modify (each node) /etc/hosts so that the localhost line only has 127.0.0.1 and does not include any machine name, such as:

127.0.0.1 localhost

And add the IPs of the nodes (and for all the nodes), for example:

192.168.0.1 pirulo1
192.168.0.2 pirulo2
...

It is possible to create a user (nteum) in all the nodes, create a group and add this user to the group:

```
groupadd beowulf
adduser nteum beowulf
echo umask 007 >> /home/nteum/.bash_profile
```

In this way, any file created by the nteum user or any within the group can be modified by the Beowulf cluster.

We must create an NFS server (and the rest of the nodes will be clients of this NFS). On the server, we create a directory as follows:

```
mkdir /mnt/nteum
chmod 770 /mnt/nteum
chown -R nteum:beowulf /mnt/nteum
```

Now we can export this directory from the server.

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

We must remember that our network will be 192.168.0.xxx and it is a private network, in other words, the cluster will not be seen from the Internet and we must adjust the configurations so that all the nodes can see each other (from the firewalls).

We should verify that the services are working:

```
chkconfig -add sshd
chkconfig -add nfs
chkconfig -add rexec
chkconfig -add rlogin
```

```
chkconfig -level 3 rsh on
chkconfig -level 3 nfs on
chkconfig -level 3 rexec on
chkconfig -level 3 rlogin on
```

To work securely, it is important to work with ssh instead of rsh, which means that we must generate the keys for interconnecting the machines-nteum user securely, without a password. To do this, we modify (we remove the comment #) the following lines in /etc/ssh/sshd_config:

RSAAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys

We reboot the machine and we connect as the nteum user, given that this user will operate the cluster. To generate keys:

```
ssh-keygen -b 1024 -f ~/.ssh/id_rsa -t rsa -N ""
```

The id_rsa and id_rsa.pub files will have been created in the /home/nteum/.ssh library directory and we must copy id_rsa.pub in a file called authorized_keys in the same directory. And we modify the permissions with `chmod 644 ~/.ssh/aut*` and `chmod 755 ~/.ssh`.

Given that only the main node will be connected to the others (and not the other way round) we only need to copy the public key (d_rsa.pub) to each node in the directory/file /home/nteum/.ssh/authorized_keys of each node. In addition, on each node, we will have to mount the NFS adding /etc/fstab the line pirulo1:/mnt/nteum /mnt/nteum nfs rw,hard,intr 0 0.

As of this point, we already have a Beowulf cluster for executing applications that could be PVM or MPI (we will see this in the following sections). Over FC, there is an application (system-config-cluster) that makes it possible to configure a cluster based on a graphic tool. For more information, please see: http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Cluster_Administration/index.html.

1.1.2. Benefits of distributed computing

What are the benefits of parallel computing? We will see these with an example [Rad]. We have a program for adding numbers (for example, 4 + 5 + 6...) called sumdis.c and written in C:

```
#include <stdio.h>

  int main (int argc, char** argv){

float initial, final, result, tmp;
```

```
if (argc < 2) {
  printf ("Use: %s N.º initial N.º final\n",argv[0]);
  exit(1);
  }
else {
  initial = atol (argv[1]);
  final = atol (argv[2]);
  result = 0.0;
  }
  for (tmp = inicial; tmp <= final; tmp++){
  result + = tmp; }
printf("%f\n", result)
return 0;
}
```

We compile it with `gcc -o sumdis sumdis.c` and if we look at the execution of this program, with, for example:

```
time ./sumdis 1 1000000 (from 1 to 106)
```

we can see that the time in a Debian 2.4.18 machine with AMD Athlon 1.400 MHz 256 Mb RAM is (approximately) real = 0,013 and user = 0,010 in other words, 13 ms in total and 10 ms inuser zone. If, however, we enter:

```
time ./sum 1 16000000 (from 1 to 16 * 106)
```

the time will be real = 182, in other words, 14 times more, which means, if we consider 160.000.000 ($160*10^6$), the time will be approximately dozens of minutes.

The idea of distributed computing is: if we have a cluster of 4 machines (node1-node4) with a server, where the file is shared by NFS, it would be interesting to divide the execution through rsh (not advisable, but it is acceptable for this example), so that the first adds from 1 to 40.000.000, the second from 40.000.001 to 80.000.000, the third from 80.000.001 to 120.000.000 and the fourth from 120.000.001 to 160.000.000. The following commands show one possibility. We consider that the system has the directory /home shared by NFS and that the user (nteum) who will execute the script has configured .rhosts adequately so that it is possible to access their account without the password. In addition, if tcpd has been activated in /etc/inetd.conf in the *rsh* line, there must be the corresponding file in /etc/hosts.allow, which would allow us to access the four machines in the cluster:

`mkfifo out`	Creates a fifo queue in /home/nteum

```
./distr.sh & time cat salida | awk '{total + = $1 } \
END printf "%lf", total}'
```

	Executes the command distr.sh; the results are collected and added whilst the execution time is measured

The shell script distr.sh can be something like:

```
rsh node1 /home/nteum/sumdis 1 40000000 > /home/nteum/out <
/dev/null &
rsh node2 /home/nteum/sumdis 40000001 80000000 > /home/nteum/
out < /dev/null &
rsh node3 /home/nteum/sumdis 80000001 120000000 > /home/
nteum/out < /dev/null &
rsh node4 /home/nteum/sumdis 120000001 160000000 > /home/
nteum/out < /dev/null &
```

We can observe that the time is significantly reduced (by a factor of approximately 4) and not exactly lineally, but almost. Obviously, this example is very simple and is only used for demonstrative purposes. The programmers use libraries that allow them to set the execution time, the creation and communication of processes in a distributed system (such as PVM and MPI).

1.2. How should we program to take advantage of concurrent computing?

There are various ways of expressing the concurrency in a program. The most common two are:

1) Using threads (or processes).

2) Using processes in different processors that communicate through messages (MPS, *message passing system*).

Both methods can be implemented on different hardware configurations (share memory or messages) but MPS systems can involve latency and speed problems with the messages on the network, which can be a negative factor. However, with the advances in network technology, these systems have grown in popularity (and in number). A message is extremely simple:

send(*destination*,*msg*)
recv(*origin*,*msg*)

The most common APIs today are PVM and MPI and, in addition, they do not limit the possibility of using threads (even if it is at a local level) or of having concurrent processing and in/out. On the other hand, on a machine with

shared memory (SHM) it is only possible to use threads and there is the severe problem of scalability, given that all the processors use the same memory and the number of processors in the system is limited by the memory's bandwidth.

To summarise, we can conclude that:

1) Proliferation of multitask (multi-user) machines connected through a network with distributed services (NFS and NIS YP).

2) They are heterogeneous systems with networked operating systems (NOS) that offer a series of distributed and remote services.

3) Distributed applications can be programmed at different levels:

a) Using a client-server model and programming at low-level (sockets).

b) The same model but with a "high-level" API (PVM, MPI).

c) Using other programming models such as programming oriented to distributed objects (RMI, CORBA, Agents...).

1.2.1. Parallel virtual machine (PVM)

PVM [Proe] is an API that makes it possible to generate, from the perspective of the application, a dynamic cluster of computers, which constitutes a virtual machine (VM). The tasks can be created dynamically (spawned) and/or eliminated (killed) and any PVM task can send a message to another. There is no limit to the size or number of messages (according to the specifications, although there may be hardware/operating system combinations that result in limitations on message size) and the model supports fault tolerance, resource control, processes control, heterogeneity in the networks and in the hosts.

The system (VM) has tools for controlling the resources (adding or deleting hosts from the virtual machine), processes control (dynamic creation/elimination of processes), different communication models (blocking send, blocking/nonblocking receive, multicast), dynamic task groups (a task can be attached or removed from a group dynamically) and fault tolerance (the VM detects the fault and it can be reconfigured).

The PVM structure is based, on the one hand, on the daemon (pvm3d) that resides in each machine and is interconnected using UDP, and, on the other hand, the PVM library (libpvm3.a), which contains all the routines for sending/receiving messages, creating/eliminating processes, groups, synchronisation etc. which will use the distributed application.

PVM has a console (pvm) that makes it possible to start up the daemon, create the VM, execute applications etc. It is advisable to install the software from the distribution, given that the compilation requires a certain amount of 'dedication'. To install PVM on Debian, for example, we must include two packages (minimum): pvm and pvm-dev (the pvm console and utilities are in the first and the libraries, header and the rest of the compiling tools are in the second). If we only need the library because we already have the application, we can install only the libpvm3 package).

To create a parallel/distributed application in PVM, we can start with the standard version or look at the physical structure of the problem and determine which parts can be concurrent (independent). The concurrent parts will be candidates for being rewritten as parallel code. In addition, we must consider whether it is possible to replace the algebraic functions with their paralleled versions (for example, ScaLapack, Scalable Linear Algebra Package, available in Debian as scalapack-pvm | mpich-test | dev, scalapack1-pvm | mpich depending on whether it is PVM or MPI). It is also convenient to find out whether there is any similar parallel application (http://www.epm.ornl.gov/pvm) that might guide us as to the construction method of the parallel application.

Parallelising a program is not an easy task, as we have to take into account Amdahl's law.

> Amdahl's law states that speedup is limited by the fraction of code (f) that can be paralleled: **speedup = 1/(1-f)**.

This law implies that a sequential application f = 0 and the speedup = 1, with all the parallel code f = 1 and speedup= infinite (!), with possible values, 90% of the parallel code means a speedup = 10 but with f = 0.99, speedup = 100. This limitation can be avoided with scalable algorithms and different application models:

Note

Amdahl's law
speedup = 1/(1-f)
f is the fraction of parallel code

1) Master-worker: the master starts up all the workers and coordinates the work and in/out.

2) Single process multiple data (SPMD): the same program that executes with different sets of data.

3) Functional: various programs that perform a different function in the application.

With the pvm console and with the add command we can configure the VM whilst adding all the nodes. In each of these nodes, there must be the directory ~/pvm3/bin/LINUX, with the binaries of the application. The variables *PVM_ROOT* = Directory must be declared, where the lib/LINUX/libpvm3.a is

and PVM_ARCH=LINUX, which can be placed, for example, in file /.cshrc. The default shell of the user (generally a NIS user or, if not, the same user must be in each machine with the same password) should be `csh` (if we use rsh as a means of remote execution) and the file /.rhosts must be configured to provide access to each node without the password. The PVM package incorporates an *rsh-pvm* that can be found in /usr/lib/pvm3/bin as an rsh specifically made for PVM (see the documentation), as there are some distributions that do not include it, for security reasons. It is advisable to configure, as we have shown, the ssh with the public keys of the server in .ssh/authorized_keys of the directory of each user.

As an example of PVM programming, we show a program of the server-client type, where the server creates the child nodes, sends the data, these nodes circulate the data a determined number of times between the child nodes (the first node on the left receives a piece of data, processes it and sends it to the one on the right), whilst the parent nodes waits for each child node to finish.

Example of PVM: master.*c*

To **compile** in Debian:
gcc -O -I/usr/share/pvm3/include/ -L/usr/share/pvm3/lib/LINUX -o master master.c -lpvm3
The directories in -I and in -L must be where the includes pvm3.h and libpvm* are located, respectively.
Execution:
1) execute the pvmd daemon with pvm
2) execute add to add the nodes (this command can be skipped if we only have one node
3) execute quit (we leave pvm but it continues to execute)
4) we execute master

Note

Compiling PVM:
gcc -O -I/usr/include/ -o output output.c -lpvm3

```
#include <stdio.h>
#include "pvm3.h"
#define SLAVENAME "/home/nteum/pvm3/client"
main() {
 int mytid, tids[20], n, nproc, numt, i, who, msgtype, loops;
 float data[10]; int n_times;

 if( pvm_parent() ==PvmNoParent ){
      /*Return if this is the parent or child process */
      loops = 1;
      printf("\n How many children (120)? ");
      scanf("%d", &nproc);
      printf("\n How many child-child communication loops (1 - 5000)? ");
      scanf("%d", &loops); }

 /*Redirects the in/out of the children to the parent */
 pvm_catchout(stdout);

 /*Creates the children */
 numt = pvm_spawn(SLAVENAME, (char**)0, 0, "", nproc, tids);
 /*Starts up a new process, 1st: executable child, 2nd: argv, 3rd :options,
```

```
4th :where, 5th :N.º copies, 6th :matrix of id*/
printf("Result of Spawn: %d \n", numt);

/*Has it managed?*/
if( numt &lt; nproc ){
 Printf("Error creating the children. Error code:\n");
 for( i = numt ; i<nproc ; i++ ) {
   printf("Tid %d %d\n",i,tids[i]); }
 for( i = 0 ; i<numt ; i++ ){
 pvm_kill( tids[i] ); }  /*Kill the processes with id in tids*/
 pvm_exit();
 exit(); /*Finish*/
 }

/*Start up parent program, initialising the data */
n = 10;
for( i = 0 ; i<n ; i++ ){
     data[i] = 2.0;}
/*Broadcast with initial data to slaves*/
pvm_initsend(PvmDataDefault);.
/*Delete the buffer and specify message encoding*/
pvm_pkint(&loops, 1, 1);
/*Package data in the buffer, 2nd N.º, 3*:stride*/
pvm_pkint(&nproc, 1, 1);
pvm_pkint(tids, nproc, 1);
pvm_pkint(&n, 1, 1);
pvm_pkfloat(data, n, 1);
pvm_mcast(tids, nproc, 0);
/*Multicast in the buffer to the tids and wait for the result from the children*/
msgtype = 5;
for( i = 0 ; i < nproc ; i++ ){
     pvm_recv( -1, msgtype );
     /*Receive a message, -1 :of any, 2nd:tag of msg*/
     pvm_upkint( &who, 1, 1 );
     /*Unpackage*/
     printf("Finished %d\n",who);
     }
pvm_exit();
}
```

Example of PVM: *client.c*

To **compile** in Debian:
gcc -O -I/usr/share/pvm3/include/ -L/usr/share/pvm3/lib/LINUX -or client client.c -lpvm3
The directories in -I and in -L must be where the included pvm3.h and libpvm* are located, respectively.
Execution:
This is not necessary as the master will start them up, but the client must be in /home/nteum/pvm3

```
#include <stdio.h>
#include "pvm3.h"

main() {
  int mytid; /*Mi task id*/
  int tids[20]; /*Task ids*/
  int n, me, i, nproc, master, msgtype, loops; float data[10];
  long result[4]; float work();
  mytid = pvm_mytid(); msgtype = 0;

  pvm_recv( -1, msgtype );
  pvm_upkint(&loops, 1, 1);
  pvm_upkint(&nproc, 1, 1);
  pvm_upkint(tids, nproc, 1);
  pvm_upkint(&n, 1, 1);
  pvm_upkfloat(data, n, 1);
  /*Determines which child it is (0 -- nproc-1) */
  for( i = 0; i < nproc ; i++ )
  if( mytid == tids[i] ){ me = i; break; }

  /*Processes and passes the data between neighbours*/
  work (me, data, tids, nproc, loops);

  /*Send the data to the master */
  pvm_initsend( PvmDataDefault );
  pvm_pkint( &me, 1, 1 );
  msgtype = 5;
  master = pvm_parent();  /*Find out who created it */
  pvm_send( master, msgtype);
  pvm_exit();
  }

float work(me, data, tids, nproc, loops)
 int me, *tids, nproc; float *data; {
  int i,j, dest; float psum = 0.0, sum = 0.1;
  for (j = 1; j <= loops; j++){
   pvm_initsend( PvmDataDefault );
   pvm_pkfloat( &sum, 1, 1 );
   dest = me + 1;
```

```
    if( dest == nproc ) dest = 0;
    pvm_send( tids[dest], 22 );
    i = me - 1;
    if (me == 0 ) i = nproc-1;
    pvm_recv( tids[i], 22 );
    pvm_upkfloat( &psum, 1, 1 );
    }
  }
```

The programmer is assisted by a graphic interface that is of great help (see following figure), which acts as a PVM console and monitor, called xpvm (in Debian XPVM, install package xpvm), which makes it possible to configure the VM, execute processes, visualise the interaction between tasks (communications), statuses, information etc.

Figure 1

1.2.2. Message passing interface (MPI)

The definition of the API of MPI [Prob, Proc] has been the work resulting from MPI Forum (MPIF), which is a consortium of more than 40 organisations. MPI has influences from different architectures, languages and works in the world of parallelism such as: WRC (Ibm), Intel NX/2, Express, nCUBE, Vertex, p4, Parmac and contributions from ZipCode, Chimp, PVM, Chamaleon, PICL. The main objective of MPIF was to design an API, without any particular relation with any compiler or library, so that efficient memory-to-memory copy communication, computing and concurrent communication and communication downloads would be possible, provided there is a communications coprocessor. In addition, it supports development in heterogeneous environments, with interface C and F77 (including C++, F90), where communication will be reliable and the faults resolved by the system. The API also needed an interface for different environments (PVM, NX, Express, p4...) and an implementation that was adaptable to different platforms with insignificant changes that did not interfere with the operating system (thread-safety). This API was designed especially for programmers that used message passing

paradigm (MPP) in C and F77 to take advantage of the most important characteristic: portability. The MPP can be executed on multiprocessor machines, WS networks and even on machines with shared memory. The MPI1 version (the most widespread version) does not support the dynamic creation (spawn) of tasks, but MPI2 (which is developing at a growing rate) does.

Many aspects have been designed to take advantage of the benefits of communications hardware on scalable parallel computers (SPC) and the standard has been mostly accepted by parallel and distributed hardware manufacturers (SGI, SUN, Cray, HPConvex, IBM, Parsystec...). There are freeware versions (mpich, for example) (which are completely compatible with the commercial implementations from the hardware manufacturers) and they include point-to-point communications, collective operations and process groups, communications and topology contexts, support for F77 and C and a control, administration and profiling environment. But there are also some unresolved points, such as: SHM operations, remote execution, program construction tools, debugging, control of threads, administration of tasks, concurrent in/out functions (most of these problems arising from a lack of tools are resolved in version 2 of API MPI2). The function in MPI1, as there is no dynamic process creation, is very simple, given that of so many processes as tasks that exist, autonomous and executing their own multiple instruction multiple data (MIMD) style code and communicating via MPI calls. The code may be sequential or multithread (concurrent) and MPI works in threadsafe mode, in other words, it is possible to use calls to MPI in concurrent threads, as the calls re-enter.

To install MPI, it is recommended that you use the distribution, given that compiling it is extremely complex (due to the dependencies that it needs from other packages). Debian includes Mpich version 1.2.7-2 (Etch) in the mpich-bin package (the mpich one is obsolete) and also mpich-mpd-bin (version of a multipurpose daemon that includes support for scalable processes, management and control). The mpich-bin implements the MPI 1.2 standard and some parts of MPI 2 (such as, for example, parallel in/out). In addition, this same distribution includes another implementation of MPI called LAM (lam* packages and documentation in /usr/doc/lam-runtime/release.html). The two implementations are equivalent, from the perspective of MPI, but they are managed differently. All the information on Mpich can be found (after installing the *mpich** packages) in /usr/share/doc/mpi (or in /usr/doc/mpi). Mpich needs rsh to execute in other machines, which means that we have to insert the user directory in a ~/.rhosts file with lines in the following format: *host username* to allow the *username* to enter the *host* without the password (the same as PVM). It should be remembered that we have to install the rshserver package on all the machines and if there is tcpd in /etc/inetd.conf on *rsh.d*, we must enable the *hosts* in /etc/hosts.allow. In addition, we must have mounted the directory of the user by NFS in all the machines and the /etc/mpich/machines.LINUX

file must contain the *hostname* of all the machines that comprise the cluster (one machine per line, by default, appears as *localhost*). In addition, the user must have the Csh as the shell by default.

On Debian, we can install the *update-cluster* package to help with the administration. The installation of Mpich on Debian uses ssh instead of rsh for security reasons, although there is a link of rsh =>ssh for compatibility. The only difference is that we must use the ssh authentication mechanisms for the connection without password through the corresponding files. Otherwise, for each process that executes, we will have to enter the password before execution. To allow the connection between machines, with ssh, without the password, we must follow the procedure mentioned in the preceding section. To check it, we can run ssh *localhost* and then we should be able to log in without the password. Bear in mind that if we install Mpich and LAM-MPI, the mpirun of Mpich will be called mpirun.mpich and the mpirun will be that of LAM-MPI. It is important to remember that mpirun of LAM will use the lamboot daemon to form the distributed topology of the VM.

The lamboot daemon has been designed so that users can execute distributed programs without having root permissions (it also makes it possible to execute programs in a VM without calls to MPI). For this reason, to execute mpirun, we will have to do it as a user other than the root and execute lamboot beforehand. lamboot uses a configuration file in /etc/lam for the default definition of the nodes (see bhost*); please consult the documentation for more information (http://www.lam-mpi.org/). [Lam]

To compile MMPI programs, we can use the mpicc command (for example, `mpicc -o test test.c`), which accepts all the options of gcc although it is advisable to use (with modifications) some of the *makefiles* that are located in the /usr/doc/mpich/examples file. It is also possible to use *mpireconfig Makefile*, that uses the Makefile.in file as an entry to generate the *makefile* and is much easier to modify. After, we can run:

```
mpirun -np 8 programme
```

or:

```
mpirun.mpich -np 8 programme
```

where np is the number of processes or processors in which the program will execute (8, in this case). We can put in the number we like, as Mpich will try to distribute the processes in a balanced manner better between all the machines of /etc/mpich/machines.LINUX. If there are more processes than processors, Mpich will use the swap characteristics of GNU/Linux to simulate

parallel execution. In Debian and in the directory /usr/doc/mpich-doc (a link to /usr/share/doc/mpich-doc), we can find all the documentation in different formats (commands, API of MPI etc.).

To compile MPI: `mpicc -O -o output output.c`

Execute Mpich: `mpirun.mpich -np Nº_processes output`

We will now see two examples (which are included in the distribution of Mpich 1.2.x in directory /usr/doc/mpich/examples). Srtest is a simple program for establishing communications between point-to-point processes and cpi calculates the value of Pi in distributed form (through integration).

Point-to-point communications: srtest.c

For compiling: mpicc -O -o srtest srtest.c
Execution of Mpich: mpirun.mpich -np *N.º_processes* srtest (will ask for the password [*N.º processes* - 1] times if we do not have direct access through *ssh*).
Execution of LAM: mpirun -np N.º_processes srtest (must be a user other than the root)

```
#include "mpi.h"
#include <stdio.h>
#define BUFLEN 512
int main(int argc, char *argv[]) {
    int myid, numprocs, next, namelen;
    char buffer[BUFLEN], processor_name[MPI_MAX_PROCESSOR_NAME];  MPI_Status status;
    MPI_Init(&argc,&argv);
        /* Must be placed before other MPI calls, always */
        MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
        MPI_Comm_rank(MPI_COMM_WORLD,&amp;myid);
        /*Integrates the process in a communications group*/
        MPI_Get_processor_name(processor_name,&namelen);
        /*Obtains the name of the processor*/
    fprintf(stderr,"Process %d on %s\n", myid, processor_name);
    strcpy(buffer,"Hello People");
    if (myid ==numprocs1) next = 0;
    else next = myid+1;
    if (myid ==0) {   /*If it is the initial, send string of buffer*/.
        printf("%d Send '%s' \n",myid,buffer);
        MPI_Send(buffer, strlen(buffer)+1, MPI_CHAR, next, 99,
          MPI_COMM_WORLD);
        /*Blocking Send, 1 or :buffer, 2 or :size, 3 or :type, 4 or :destination, 5
        or :tag, 6 or :context*/
        /*MPI_Send(buffer, strlen(buffer)+1, MPI_CHAR,
        MPI_PROC_NULL, 299,MPI_COMM_WORLD);*/
        printf("%d receiving \n",myid);
        /* Blocking Recv, 1 o :buffer, 2 or :size, 3 or :type, 4 or :source, 5
        or :tag, 6 or :context, 7 or :status*/
        MPI_Recv(buffer, BUFLEN, MPI_CHAR, MPI_ANY_SOURCE, 99, MPI_COMM_WORLD,&status);
```

```
        printf("%d received '%s' \n",myid,buffer) }
    else {
        printf("%d receiving \n",myid);
        MPI_Recv(buffer, BUFLEN, MPI_CHAR, MPI_ANY_SOURCE, 99, MPI_COMM_WORLD,status);
        /*MPI_Recv(buffer, BUFLEN, MPI_CHAR, MPI_PROC_NULL, 299,MPI_COMM_WORLD,&status);*/
        printf("%d received '%s' \n",myid,buffer);
        MPI_Send(buffer, strlen(buffer)+1, MPI_CHAR, next, 99,
        MPI_COMM_WORLD);
        printf("%d sent '%s' \n",myid,buffer);}
    MPI_Barrier(MPI_COMM_WORLD); /*Synchronises all the processes*/   MPI_Finalize();
    /*Frees up the resources and ends*/ return (0);
  }
```

Calculation of distributed PI: cpi.c

For compiling: *mpicc O* or *cpi cpi.c*.
Execution of Mpich: mpirun.mpich -np N.º processes cpi (will ask for the password (N.° processes - 1) times if we do not have direct access through *ssh*).
Execution of LAM: mpirun -np N.º processes cpi (must be a user other than root).

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>
double f( double );
double f( double a) { return (4.0 / (1.0 + a*a)); }
int main( int argc, char *argv[] ) {
    int done = 0, n, myid, numprocs, i;
    double PI25DT = 3.14159265358979323846264 3;
    double mypi, pi, h, sum, x;
    double startwtime = 0.0, endwtime;
    int namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&amp;numprocs);
          /*Indicates the number of processes in the group*/
          MPI_Comm_rank(MPI_COMM_WORLD,&myid);
          /*Id of the process*/   MPI_Get_processor_name(processor_name,&namelen);
          /*Name of the process*/
    fprintf(stderr,"Process %d on %s\n", myid, processor_name);
    n = 0;
    while (!done) {
        if (myid ==0) { /*If it is the first...*/
          if (n ==0) n = 100; else n = 0;
          startwtime = MPI_Wtime();} /* Time Clock */
        MPI_Bcast(&amp;n, 1, MPI_INT, 0, MPI_COMM_WORLD);    /*Broadcast to the rest*/
        /*Send from 4th arg. to all
        the processes of the group. All others that are not 0
        will copy the buffer from 4 or arg -process 0-*/ /*1.º:buffer,
```

```
          2nd :size, 3rd :type, 5th :group */
          if (n == 0) done = 1; else {
              h = 1.0 / (double) n;
              sum = 0.0;
              for (i = myid + 1; i &lt;= n; i + = numprocs) {
                x = h * ((double)i - 0.5); sum + = f(x); }
              mypi = h * sum;
          MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0,
          MPI_COMM_WORLD);
          /* Combines the elements of the Send Buffer of each process of the
          group using the operation MPI_SUM and returns the result in
          the Recv Buffer. It must be called by all the processes of the
          group using the same arguments*/ /*1st :sendbuffer, 2nd
          :recvbuffer, 3rd :size, 4th :typo, 5th :oper, 6th :root, 7th
          :context*/
          if (myid == 0){ /*Only the P0 prints the result*/
          printf("Pi is approximately %.16f, the error is %.16f\n", pi, fabs(pi - PI25DT));
          endwtime = MPI_Wtime();
          printf("Execution time = %f\n", endwtime-startwtime); }
          }
    }
    MPI_Finalize(); /*Free up resources and finish*/
    return 0;
  }
```

As XPVM exists in PVM, in MPI there is an analogous application (more sophisticated) called XMPI (xmpi in Debian). It is also possible to install a library, libxmpi3, which implements the XMPI protocol to graphically analyse MPI programs with more details than offered in xmpi. The following figure shows some of the possible graphics in xmpi.

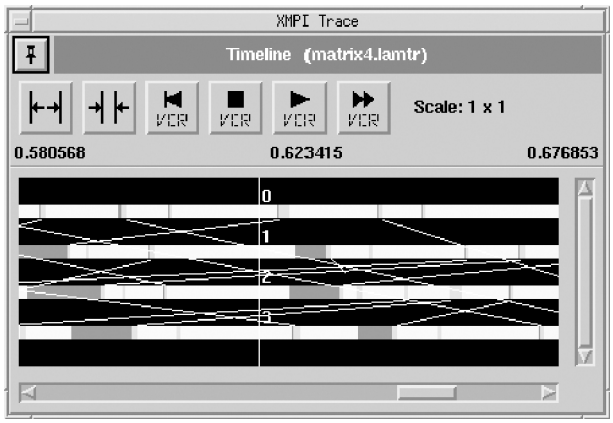

Figure 2. XMPI

2. OpenMosix

OpenMosix [Prod] is a software package that transforms a set of machines connected by a network under GNU/Linux in a cluster. This balances the workload automatically between the different nodes of the cluster and the nodes can be joined or the cluster left without interrupting the service. The load is distributed between the nodes, taking into account the speed of the connection and the CPU. OpenMosix is part of the kernel (through a Linux Kernel Patch) and maintains total compatibility with GNU/Linux, the user programs, files and resources. Another characteristic of OpenMosix is that it incorporates a powerful and optimised file system (oMFS) for HPC (high performance computing) applications. In Debian Woody, we can install OpenMosix from openmosix-dev (libraries and headers), kernel-pacth- openmosix (OpenMosix patch), openmosix (administration tools). Likewise, it is possible to install mosix (see the documentation for the difference, especially with regard to the licenses, between Mosix and OpenMosix). In Debian versions subsequent to Woody, it is not included as a package (stable) and it will be necessary to go to http://openmosix.sourceforge.net/ to obtain the packages (or resources) and the installation guides (http://howto.x-tend.be/openMosix-HOWTO/).

OpenMosix uses a configuration file that is generally found in en /etc (see documentation for older versions of this file), which is called openmosix.map and which should be in each node. Its format is very simple and each line has three fields: Nodo_ID IP-Address(or hostname) Range-size

An example would be:

1 node1 1
2 node2 1
3 node3 1
4 192.168.1.1 1
5 192.168.1.2 1

It is also possible to use a range where the ID and the IP increase respectively. We have to ensure that we have the same configuration and the same version of OpenMosix in each node. To execute OpenMosix, in each node we must type:

```
setpe -w -f /etc/openmosix.map
```

We can also use the OpenMosix script (copying it from userspace-tools to /etc/init.d) to start it up during boot.

The oMFS file system permits remote access to all the files in the cluster, as though they were locally mounted. The file systems (FS) of the other nodes can be mounted on /mfs and, therefore, the files in /home on node 3 will be seen on each machine in /mfs/3/home.

All the UIDs (User IDs) and GIDs (Group IDs) of the FS on each node of the cluster must be equal (OpenLdap could be used for this).

To mount the oMFS, we must modify /etc/fstab with an entry such as: mfs_mnt /mfs mfs dfsa = 1 0 0 and to enable or disable it: mfs_mnt /mfs mfs dfsa = 0 0 0.

Afterward, the FS of each node will be seen in mfs/[openMosixNode ID]/. Once installed, it will be possible to execute a very simple script various times, such as, for example (see *Howto* of OpenMosix):

```
awk 'BEGIN {for(i = 0;i<10000;i++)for(j = 0;j<10000;j++);}'
```

And, subsequently, we can observe the behaviour with `mosmom` or with `openmosixview` (recommended). OpenMosix has a daemon (`omdiscd`), which makes it possible to automatically configure the cluster eliminating the need to edit and configure /etc/openmosix.map. This daemon uses multicast to indicate the other nodes that it is also an OpenMosix node, which means that, once omdiscd has booted, this daemon will join the cluster automatically. For this to happen, we need to have the default routing (GW) of the network properly configured. Once it has executed (omdiscd), a series of messages indicating the status of the cluster and the configuration will be generated. We can use the showmap command to see the new configuration generated by omdiscd. OpenMosix provides a set of tools that the administrator can use to configure and tune the OpenMosix cluster. These tasks can be performed with tools in the space of a user (*migrate, mon, mosctl, mosrun*) or through the /proc/hpc interface. It is important to remember that up to OpenMosix version 2.4.16, the interface was called /proc/mosix and that, since version 2.4.17, it has been called /proc/hpc.

We will now present a summary of the configuration tools that are executed in the space of a user; for /proc/hpc consult the references:

- `migrate` [PID] [OpenMosix ID]: sends a migration request to a process.

- `mon`: is a monitor with a text interface that shows information on the cluster through a bar diagram.

- `mosctl`: is the configuration tool of OpenMosix. Using the options (stay, lstay, block, quiet, mfs, expel, bring, get- tune, getyard, getdecay) we can

indicate whether processes can migrate or not, the use of MFS, obtain information on the load, balance on the load etc.

- `mosrun` [h | OpenMosix ID | list of OpenMosix IDs] command [arguments]: executes a command on a determined node.

3. Metacomputers, grid computing

The computing requirements that are needed for certain applications are so large that they require thousands of hours to be able to execute in cluster environments. Such applications have promoted the creation of virtual computers on networks, metacomputers or grid computers. This technology has made it possible to connect execution environments, high-speed networks, databases, instruments etc., distributed in different geographic locations. This makes it possible to achieve a processing power that would not be economically viable in any other way with excellent results. Examples of their application are experiments such as the I-WAY networking (which connects supercomputers from 17 different places) in North America, or DataGrid, CrossGrid in Europa or IrisGrid in España. These metacomputers or grid computers have a lot in common with parallel and distributed systems (SPD), but they are also different in certain important aspects. Although they are connected through networks, the networks can have different characteristics, the service cannot be guaranteed and they will be located in different domains. The programming model and interfaces must be radically different (in respect of the model of distributed systems) and adequate for high performance computing. As with SPD, the metacomputing applications require a communications plan to provide the required performance levels; but given their dynamic nature, new tools and techniques are needed. In other words, whilst metacomputing can be formed with the base of the SPDs, it is necessary to create new tools, mechanisms and techniques for these. [Fos]

3.1. Different computing architectures

If we only consider the calculative power aspect, we can see that there are various solutions depending on the size and characteristics of the problem. Firstly, we could think of a supercomputer (server) but these have problems such as the lack of scalability, costly equipment and maintenance, peak computing (a lot of time resources are not taken advantage of) and reliability problems. The economic alternative is a set of computers interconnected by a high performance network (Fast Ethernet – LAN – or Myrinet – SAN) which would form a cluster of stations dedicated to parallel/distributed computing (SPD) with a very high performance level (3 to 15 times cost/performance ratio). But these systems have inconveniences such as the high cost of communications, maintenance, programming model etc. However, it is an excellent solution for medium range or high time computing (HTC). Another interesting concept is intranet computing, which means using the equipment of a local network (for example, a C class network) to execute sequential or parallel jobs with assistance of an administration and load tool; In other words, it is a step down from a cluster and it permits the exploitation of the computational power in a large local network with the ensuing advantages, as we increase the effec-

tiveness of the use of resources (low cost CPU cycles), improve the scalability and the administration is not too complex. For these types of solutions, there is software such as Sun Grid Engine by Sun Microsystems [Sun], Condor by the University of Wisconsin (both free) [Uni] or LSF by Platform Computing (commercial) [Pla].

The option of intranet computing presents some inconveniences such as the impossibility of managing resources outside the domain of administration. Some of the abovementioned tools (Condor, LSF or SGE) permit cooperation between different sub-nodes of the system, but all of them must have the same administrative structure, the same security policies and the same philosophy of resource management. Although this is a step forward in terms of computational power at low-cost, they only manage the CPU and not the data shared between the sub-nodes. Besides, the protocols and interfaces are proprietary and they are not based on an open standard, it is not possible to amortise the resources when they are not fully in use and neither can we share resources with other organisations. [Beo, Ext, Die]

The growth of computers between 1986 and 2000 has multiplied by 500 and the networks by 340,000, but forecasts would indicate that, between 2001 and 2010, computers will only multiply by 60 and networks by 4,000. This indicates the standard of the next architecture for HPC: computing distributed by Internet or grid computing (GC) or metacomputing.

Grid computing is a new emerging technology, the objective of which is to share resources by Internet in a uniform, transparent, secure, efficient and reliable manner. This technology is complementary to the preceding technologies, in that it permits the interconnection of resources in different administration domains whilst respecting their internal security policies and their resource management software on the intranet. According to one of its precursors, Ian Foster, in his article "What is the Grid? A Three Point Checklist" (2002), a grid is a system that:

1) coordinates resources that are not subject to centralised control,

2) using standard, open, general-purpose protocols and interfaces,

3) to deliver non-trivial qualities of service.

Among the advantages that this new technology provides, we might mention the lease of resources, the amortisation of own resources, a great amount of power without having to invest in resources and installations, collaboration/sharing between institutions and virtual organisations etc.

The following figure provides a view of all these concepts. [Llo]

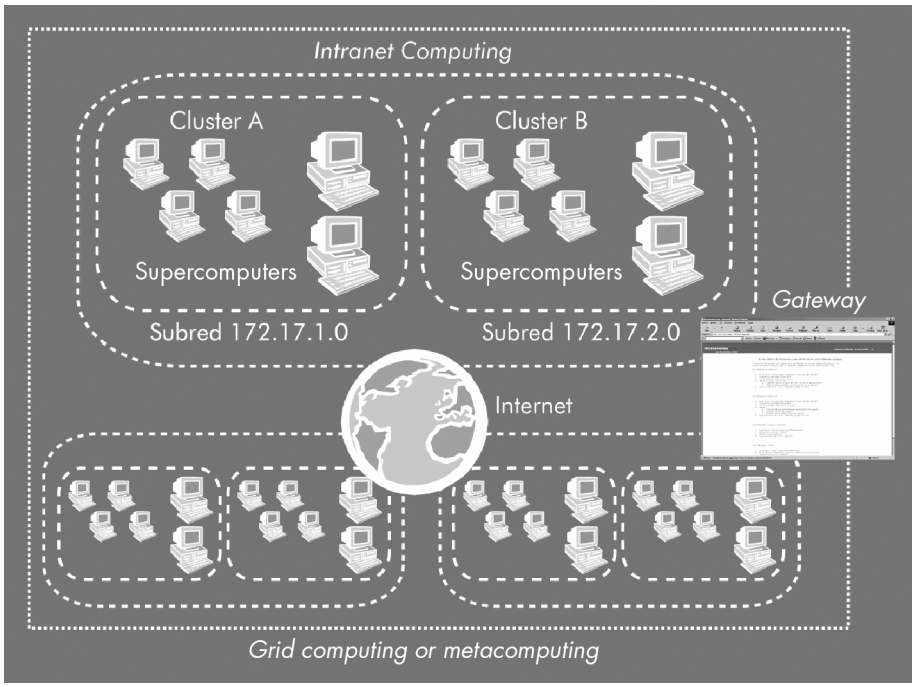

Figure 3

3.2. Globus

The Globus Project[Gloa, Glob] is one of the most emblematic in this sense, as it is the precursor in the development of a toolkit for metacomputing or grid computing and it provides considerable advances in the areas of communication, information, location and planning of resources, authentication and access to data. In other words, Globus makes it possible to share resources located in different administration domains, with different security and resource management policies and it is formed by a middleware software package that includes a set of libraries, services and API.

The *globus* tool (*Globus toolkit*) is formed by a set of modules with well-defined interfaces for interacting with other modules and/or services. The functions of these modules are as follows:

- Location and allocation of resources; this allows us to tell the applications what the requirements are and the resources that we need, given that an application cannot know where the resources on which it will execute are located.

- Communications; this provides the basic communication mechanisms, which represent an important aspect of the system, as they have to allow various methods for the applications to use them efficiently. These include message passing, remote procedure calls (RPC), shared distributed memory, (stream-based) dataflow and multicast.

- *Unified resource information service* provides a uniform mechanism for obtaining information in real time as to the status and structure of the metasystem where the applications are executing.

- Authentication interface; these are the basic authentication mechanisms for validating the identity of the users and resources. The module generates the upper layer that will then use the local services for accessing the data and resources of the system.

- Creation and execution of processes; this is used to start the execution of tasks that have been allocated to the resources, transmitting the execution parameters and controlling them until execution is completed.

- Data access; this has to provide high-speed access to the data saved in the files. For DB, it uses distributed access technology or through CORBA and it is able to achieve optimal performance levels when it accesses parallel file systems or in/out devices through the network, such as high performance storage system (HPSS).

The internal structure of Globus can be seen in the following figure (http://www.globus.org/toolkit/about.html).

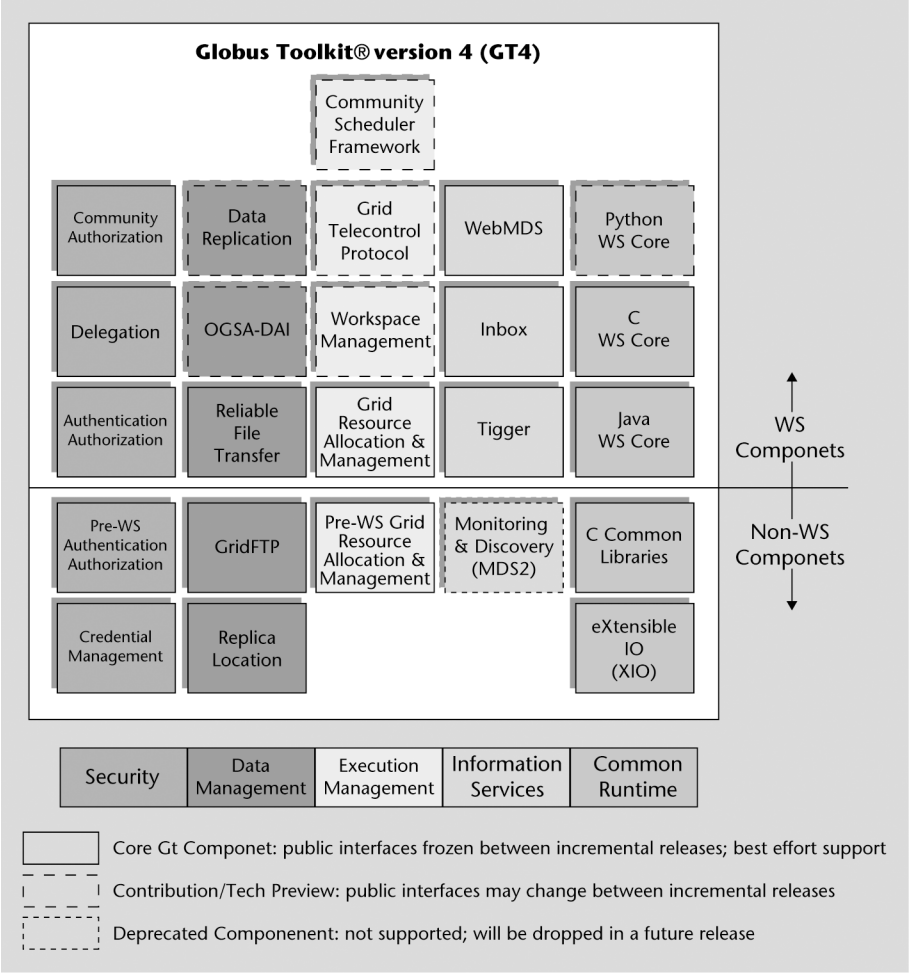

Figure 4

3.3. Software, installation and administration of Globus

The 'The Globus Alliance' website is http://www.globus.org [Gloa]. Here we can find source code and all the documents that we might need to transform our intranet into a part of a grid. Being part of a grid means agreeing to and implementing the policies of all the institutions and companies that are part of that grid. There are various different initiatives based on Globus in Spain. One of these is IrisGrid [Llo], which we can join if we wish to take advantage of the benefits of this technology. For more information, see: http://www.rediris.es/irisgrid/.

The first step for setting up Globus is to obtain the software (currently Globus Toolkit 4) called GT4. This software implements the services with a combination of C and Java (the C components can only be executed in UNIX GNU/Linux platforms, generally), which is why the software is divided into the services that it offers. Certain packages, or others, should be installed, depending on the system that we wish to set up.

A quick installation guide, with the download, system requirements and certificates can be found at http://www.globus.org/toolkit/docs/4.0/admin/docbook/quickstart.html. To summarise, the following steps must be taken:

1) Pre-requisites: verify the software and versions (zlib, j2se, disable gcj, apache, C/C++, tar, make, sed, perl, sudo, postgres, iodbc)
2) Create user, download and compile GT4
3) Start up system security (certificates)
4) Start up GridFTP
5) Start up the Webservices Container
6) Configure RFT (Reliable File Transfer)
7) Start up WS GRAM (job management)
8) Start up the second machine
9) Start up the Index Service hierarchy
10) Start up the cluster
11) Establish Cross-CA Trust

As you will observe, installing and setting up GT4 is not an easy task, but it is justified if we wish to incorporate a cluster into a *grid* or if we wish to perform tests (we recommend an extra dose of enthusiasm and patience) to appreciate the real power of GT4. For detailed information on installing GT4, please see:

http://www.globus.org/toolkit/docs/4.0/admin/docbook/

Activities

1) Install PVM on a node and execute the program master.c and client.c given as examples and observe their behaviour through xpmv.

2) Install and configure Mpich on a node; compile and execute the program cpi.c.

3) Install and configure LAM-MPI on a node; compile and execute the program cpi.c. and observe the behaviour through xmpi.

Bibliography

Other sources of reference and information

[Debc, Ibi, Mou01]

Lam-mpi: http://www.lam-mpi.org/

System-config-cluster (FC): http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Cluster_Administration/index.html

OpenMosix: http://openmosix.sourceforge.net/

HowTo Openmosix: http://howto.x-tend.be/openMosix-HOWTO/

Globus4: http://www.globus.org/toolkit/docs/4.0/

GT4 Quick Guide: http://www.globus.org/toolkit/docs/4.0/admin/docbook/quickstart.html

Bibliography

[Aiv02] **Tigran Aivazian** (2002). "Linux Kernel 2.4 Internals". *The Linux Documentation Project* (guías).

[Ano99] **Anonymous.***Maximum Linux Security: A Hacker's Guide to Protecting*

[Apa] Apache2 + SSL.
<http://www.debian-administration.org/articles/349>

[Apab] Apache2 + WebDav
<http://www.debian-administration.org/articles/285>

[Apac] Apache2 + Subversion
<http://www.debian-administration.org>

[Ar01] **Jonathan Corbet; Alessandro Rubini**. *Linux Device Drivers 2nd Editon.* O'Reilly, 2001.

[Arc] **Roberto Arcomano.** "Kernel Analysis-HOWTO". *The Linux Documentation Project.*

[Aus] **CERT Australia.** "Australian CERT".
<http://www.auscert.org.au/>

[Bac86] **Maurice J. Bach** (1986). *The Design of the UNIX Operating System.* Prentice Hall.

[Bai03] **Edward C. Bailey** (2003). *RedHat Maximum RPM.*
<http://www.redhat.com/docs/books/max-rpm/index.html>

[Ban] **Tobby Banerjee.** "Linux Installation Strategies HOWTO". *The Linux Documentation Project.*

[Bar] **Slashdot**. *slashdot site.*
<http://barrapunto.com>

[Bas] **Mike G.** "BASH Programming - Introduction HOWTO". *The Linux Documentation Project.*

[Beo] **Beowulf.org**. *Beowulf Web Site.*
<http://www.beowulf.org>

[Bor] **Matthew Borowski** (2000). "FTP". *The Linux Documentation Project.*

[Bro] **Scott Bronson** (2001). " VPN PPP-SSH". *The Linux Documentation Project.*

[Bul] **Bulma.** "Bulma Linux User Group".
<http://bulmalug.net>

[Bur02] **Hal Burgiss** (2002). "Security QuickStart HOWTO for Linux". *The Linux Documentation Project.*

[Cac] Monitoring with Cacti.

<http://cacti.net/>

[Cdg] **Cedega.** (Environment for portability of GNU/Linux games)
<http://www.transgaming.com/>

[Ced] **Cederqvist.** "Version Management with CVS".
<http://www.cvshome.org>

[Cen] The Comunity ENTerprise Operatyng System
<http://www.centos.org>

[CERa] **CERT.** "CERT site".
<http://www.cert.org>

[CERb] **CERT** (2003). "CERT vulnerabilities".
<http://www.cert.org/nav/index red.html>

[Cerc] **Cervisia.** "Cervisia interface for CVS".
<http://cervisia.sourceforge.net>

[Cis00] **Cisco** (2000). "TCP/IP White Paper".
<http://www.cisco.com>

[Com01] **Douglas Comer** (2001). *TCP/IP Basic principles, protocols and architecture* . Prentice Hall.

[Coo] **Mendel Cooper** (2006). "Advanced bashScripting Guide". *The Linux Documentation Project* (guías).

[CVS] **CVShome.org.** "CVS Home".
<http://www.cvshome.org>

[CVSI] Graphic interfaces for CVS
<http://www.twobarleycorns.net/tkcvs.html>

[DBo] **Marco Cesati; Daniel Bovet** (2006). *Understanding the Linux Kernel* (3.ª ed.). O'Reilly.

[Deb] **Debian.** "Debian Security Site".
<http://www.debian.org/security/>

[Deb04] **Debian** (2004). "APT-HOWTO".
<http://www.debian.org/doc/manuals/apt-howto/index.en.html>

[Deba] **Debian.** "Free Software vs Open Software".
<http://www.debian.org/intro/free.es.html>

[Debb] **Comunidad Debian.** "Debian Distribution".
<http://www.debian.org>

[Dieb] **Hank Dietz** (2004). "Linux Parallel Processing". *The Linux Documentation Project.*

[Dis] **Distrowatch.** "Available Linux distributions".
<http://www.distrowatch.com>

[Dgn] The Dot Gnu Project.
<http://www.gnu.org/software/dotgnu/>

[DNS] Start up a DNS Server.
<http://tldp.org/HOWTO/DNS-HOWTO-7.html>

[Dra] **Joshua Drake** (1999). "Linux Networking". *The Linux Documentation Project.*

[DSL] **Digital Line Subscriber** (2002). *The Linux Documentation Project.*

[Buy] **Kris Buytaert and others** (2002). "The OpenMosix". *The Linux DocumentationProject.*

[Ext] **ExtremeLinux.org.** "Extreme Linux Web Site".
<http://www.extremelinux.org>

[Exim] **Exim.** Mail service (MTA).

<http://www.exim.org/docs.html>

[FBI] FBI. "FBI Brigade for cybercrime".
<http://www.emergency.com/fbi-nccs.htm>

[Fed] The Fedora Project.
<http://fedoraproject.org>

[Fen02] **Kevin Fenzi.** "Linux security HOWTO". *The Linux Documentation Project.*

[Fos] **Ian Foster; Carl Kesselmany** (2003). "Globus: A Metacomputing Infrastructure Toolkit".
<http://www.globus.org7gt;

[Fre] **Freshmeat.** "Freshmeat site".
<http://freshmeat.org>

[Fri02] **Aleen Frisch** (2002). *Essential System Administration.* O'Reilly.

[Fry] Monitoring with Frysk.
<http://sources.redhat.com/frysk/>

[FSF] **FSF.** "Free Software Foundation and GNU Project".
<http://www.gnu.org>

[Gar98] **Bdale Garbee** (1998). *TCP/IP Tutorial.* N3EUA Inc.

[Gloa] **Globus. GT4.** "Admin Guide Installation" and "Admin Guide Configuration".
<http://www.globus.org>

[Glob] **Globus.** "User's Guide Core Framework Globus Toolkit ",
<http://www.globus.org>

[Gt] **Dirk Allaert Grant Taylor.** "The Linux Printing HOWTO". *The Linux Documentation Project.*

[GT4] Quick Guide.
<http://www.globus.org/toolkit/docs/4.0/admin/docbook/quickstart.html>

[Gnu] **Gnupg.org**. *GnuPG Web Site.*
<http://www.gnupg.org/>

[Gon] **Guido Gonzato.** "From DOS/Windows to Linux HOWTO". *The Linux Documentation Project.*

[Gor] **Paul Gortmaker** (2003). "The Linux BootPrompt HOWTO". *The Linux Documentation Project.*

[Gre] **Mark Grennan.** "Firewall and Proxy Server HOWTO". *The Linux Documentation Project.*

[Hat01] **Brian Hatch** (2001). *Hacking Linux Exposed.* McGraw-Hill.

[Hat03] **Red Hat** (2003). "Firewalls" en Red Hat 9 manual.
<http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/security-guide/ch-fw.html#S1-FIREWALL-IPT>

[Hatb] **Red Hat** (2003). "Red Hat 9 Security Guide".
<http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/security-guide/>

[Hatc] **Red Hat** (2003). "Red Hat Security Site".
<http://www.redhat.com/security/>

[Hatd] **Red Hat** (2003). *Use of GPG signatures in Red Hat.*
<http://www.redhat.com/docs/manuals/linux/RHL-7.3-Manual/custom-guide/ch-gnupg.html>

[Hen03] **Bryan Henderson.** "Linux Loadable Kernel Module HOWTO". *The Linux Documentation Project.*

[Him01] **Pekka Himanen** (2001). *Hacker ethics and the spirit of the information age* . Destination.

[Hin00] **Martin Hinner.** "Filesystems HOWTO". *The Linux Documentation Project.*

[His] **HispaLinux.** "Linux Hispanic Community".
<http://www.hispalinux.es>

[IET] **IETF.** "Request For Comment Repository developed by the Internet Engineering Task Force (IETF) in the Network Information Center (NIC)".
<http://www.cis.ohio-state.edu/rfc/>

[Ian] **Iana.** "List of TCP/IP ports".
<http://www.iana.org/assignments/port-numbers>

[IP] Routing with the ip tool. <ftp://ftp.inr.ac.ru/ip_routing/>

[ipw] Firmware for wireless cards IPW2200.
<http://ipw2200.sourceforge.net/firmware.php>

[Ibi] **Ibiblio.org** (2003). "Linux Documentation Center".
<http://www.ibiblio.org/pub/Linux/docs/HOWTO/>

[Incb] **Incidents.org.** "vulnerabilities Incidents".
<http://isc.incidents.org>

[Ins] **Insecure.org** (1998). "Vulnerabilities and exploits".
<http://www.insecure.org/sploits.html>

[Insa] **Insecure.org.** "Insecure.org site".
<http://www.insecure.org>

[Insb] **Insecure.org** (2003). "Nmap".
<http://www.insecure.org/nmap/index.html>

[Log] LogCheck.
<http://logcheck.org/>

[LWP] LWP: Apache+MySQL+:PHP.
<http://www.lawebdelprogramador.com/temas/tema_stablephpapachemysql.php>

[Joh98] **Michael K. Johnson** (1998). "Linux Information Sheet". *The Linux Documentation Project.*

[Jou] **Linux Journal.***Linux Journal [Linux Magazine].*
<http://www.linuxjournal.com>

[Kan] **Ivan Kanis.** "Multiboot with GRUB Mini-HOWTO". *The Linux Documentation Project.*

[Kat] **Jonathan Katz.** "Linux + Windows HOWTO". *The Linux Documentation Project.*

[KD00] **Olaf Kirch; Terry Dawson**. *Linux Network Administrator's Guide*. O'Reilly Associates. And how *e-book* (free) in Free Software Foundation, Inc., 2000.
<http://www.tldp.org/guides.html>

[Ker02] **Kernelhacking.org** (2002). "Kernel Hacking Doc Project".
<http://www.kernelhacking.org>

[Kera] **Kernelnewbies.org.** "Kernel Newbies".
<http://www.kernelnewbies.org>

[Kerb] **Kernel.org.** "Linux Kernel Archives".
<http://www.kernel.org>

[Kie] **Robert Kiesling** (1997). "The RCS (Revision Control System)". *The Linux Documentation Project.*

[Knp] Knoppix Distribution.
<http://knoppix.org>

[Koe] Kristian Koehntopp. "Linux Partition HOWTO". *The Linux Documentation Project.*

[Kuk] **Thorsten Kukuk** (2003). "The Linux NIS(YP)/NYS/NIS+". *The Linux Documentation Project.*

[Lam] **LamMPI.org.** "LAM (Local Area Multicomputer)".
<http://www.lam-mpi.org>

[Law07] **David Lawyer** (2007). "Linux Modem". *The Linux Documentation Project.*

[Lev02] **Bozidar Levi** (2002). *UNIX administration.* CRC Press.

[Lev] **Eric Levenez.** "UNIX History".
<http://www.levenez.com/unix>

[Lin03b] *FHS Standard,* 2003.
<http://www.pathname.com/fhs>

[Linc] *Linux Standards Base project.*
<http://www.linux-foundation.org/en/LSB>

[Line] **Linuxsecurity.com**. *Linux Security Reference Card.*
<http://www.linuxsecurity.com/docs/QuickRefCard.pdf>

[lkm] **lkml.***Linux Kernel Mailing List.*
<http://www.tux.org/lkml>

[Llo] **Ignacio Martín Llorente**. *State of Grid Technology and IrisGrid Initiative .*
<http://www.rediris.es/irisgrid>

[Lan] **Nicolai Langfeldt; Jamie Norrish** (2001). "DNS". *The Linux Documentation Project.*

[Log] **Logcheck.** "Logckeck Web Site".
<http://logcheck.org/>

[LPD] **LPD**. *The Linux Documentation Project.*
<http://www.tldp.org>

[Mag] **Linux Magazine**. *Linux Magazine.*
<http://www.linux-mag.com/>

[Maj96] **Amir Majidimehr** (1996). *Optimizing UNIX for Performance.* Prentice Hall.

[Mal96] **Fred Mallett** (1996). *TCP/IP Tutorial.* FAME Computer Education.

[Mal07] **Luiz Ernesto Pinheiro Malère** (2007). "Ldap". *The Linux DocumentationProject.*

[Miq] **Miquel, S.** "NIS Debian". *On Debian Woody,* /usr/doc/nis/ nis.debian.howto.

[Moin] Moin Moin
<http://moinmoin.wikiwikiweb.de/>

[Moi] Moin Moin + Debian.
<http://moinmoin.wikiwikiweb.de/MoinMoinPackages/DebianLinux>

[Mon] Monit.
<http://www.tildeslash.com/monit/>

[Monb] Monitoring with Munin and monit.
<http://www.howtoforge.com/server_monitoring_monit_munin>

[Monc] Monitoring with SNMP and MRTG.
<http://www.linuxhomenetworking.com/wiki/index.php/
>Quick_HOWTO_:_Ch22_:_Monitoring_Server_Performance

[Mono] Mono project.
<http://www.mono-project.com/Main_Page>

[Mor03] **Daniel Morill** (2003). *Configuration of Linux systems .* Anaya Multimedia.

[Mou01] **Gerhard Mourani** (2001). *Securing and Optimizing Linux: The Ultimate Solution.*
Open Network Architecture, Inc.

[Mun] Munin.
<http://munin.projects.linpro.no/>

[MRTG] MRTG.

<http://oss.oetiker.ch/mrtg/>

[Mur] **Gary Lawrence Murphy**. *Kernel Book Project.*
<http://kernelbook.sourceforge.net>

[Mutt] Mutt mail client.
<http://www.mutt.org>

[Mys] **Mysql.** "Reference Manual".
<http://www.mysql.com/>

[MysqlA] **Mysql Administrator.**
<http://www.mysql.com/products/tools/administrator/>

[Nes] **Nessus.org.** "Nessus".
<http://www.nessus.org>

[Net] **Netfilter.org**. *Netfilter/IPtables Project.*
<www.netfilter.org>

[Neu] **Christopher Neufeld.** "Setting Up Your New Domain Mini-HOWTO". *The Linux Documentation Project.*

[New] **Newsforge.** "Newsforge site".
<http://newsforge.org>

[NIS] Setting up a NIS Server.
<http://tldp.org/HOWTO/NIS-HOWTO/verification.html>

[NSAa] **NSA.** "NIST site".
<http://csrc.nist.gov/>

[NSAb] **NSA** (2003). "Security Enhanced Linux".
<http://www.nsa.gov/selinux>

[Nt3] NTFS-3g Project: NTFS-3G Read/Write Driver.
<http://www.ntfs-3g.org/>

[Oke] **Greg O'Keefe.** "From Power Up To bash Prompt HOWTO". *The Linux Documentation Project.*

[Open] **OpenVPN.** Virtual private network.
<http://openvpn.net/howto.html>

[OpenM] OpenMosix.
<http://openmosix.sourceforge.net/>>

[OpenMb] HowTo Openmosix.
<http://howto.x-tend.be/openMosix-HOWTO/>

[OSDa] **OSDL.** "Open Source Developement Laboratories".
<http://www.osdl.org>

[OSDb]OSDN. "Open Source Development Network".
<http://osdn.com>

[OSIa] **OSI.** "List of Open Source licenses".
<http://www.opensource.org/licenses/index.html>

[OSIb] **OSI** (2003). "Open Source Definition".
<http://www.opensource.org/docs/definition.php>

[OSIc] **OSI** (2003). "Open Source Iniciative".
<http://www.opensource.org>

[Peñ] **Javier Fernández-Sanguino Peña** (2007). "Securing Debian Manual".
<http://www.debian.org/doc/manuals/securing-debian-howto/>

[Pga] **PgAccess.** Client for PostgreSQL.
<http://www.pgaccess.org/>

[Pla] **Plataform.** "LSF".

<http://www.platform.com>

[Posa] **PostgreSQL.org.** "PostgreSQL Administrator's Guide".
<http://www.postgresql.org/docs/>

[Per] Performance Monitoring Tools for Linux.
<http://www.linuxjournal.com/article.php?sid=2396>

[Pose] **PostgreSQL.org.** "PostgreSQL Web Site".
<http://www.postgresql.org>

[PPP] **Linux PPP** (2000). "Corwin Williams, Joshua Drake and Robert Hart". *The Linux Documentation Project.*

[Pra03] **Joseh Pranevich** (2003). "The Wonderful World of Linux 2.6".
<http://www.kniggit.net/wwol26.html>

[Pri] **Steven Pritchard.** "Linux Hardware HOWTO". *The Linux Documentation Project.*

[Pro] **GNU Project.** "Grub Manual".
<http://www.gnu.org/software/grub/manual/>

[Proa] **Bastille Project.** "Bastille".
<http://bastille-linux.sourceforge.net/>

[Prob] **Mpich Project.** "MPI".
<http://www.mcs.anl.gov:80/mpi/>

[Proc] **Mpich Project.** "Mpich MPI Freeware".
<http://www-unix.mcs.anl.gov/mpi/>

[Prod] **OpenMosix Project.** "OpenMosix".
<http://openMosix.sourceforge.net>

[Proe] **PVM Project.** "PVM Web Site".
<http://www.csm.ornl.gov/pvm/>

[Proc] ProcMail.
<http://www.debian-administration.org/articles/242>

[ProX]Proxy Cache.
<http://www.squid-cache.org/>

[ProT] Transparent Proxy.
<http://tldp.org/HOWTO/TransparentProxy-1.html>

[Prof] ProFTP: FTP file server.
<http://www.debian-administration.org/articles/228>

[PS02] **Ricardo Enríquez Pio Sierra** (2002). *Open Source*. Anaya Multimedia.

[PurF] PureFTP: FTP file server.
<http://www.debian-administration.org/articles/383>

[Qui01] **Ellie Quigley** (2001). *Linux shells by Example*. Prentice Hall.

[Ran] **David Ranch** (2005). "Linux IP Masquerade" and **John Tapsell**. *Masquerading Made Simple*. The Linux Documentation Project.

[Ray98] **Eric Raymond** (1998). "The cathedral and the bazaar".
<http://es.tldp.org/Otros/catedral-bazar/cathedral-es-paper-00.html>

[Ray02a] **Eric Raymond** (2002). "UNIX and Internet Fundamentals". *The LinuxDocumentation Project.*

[Rayb] **Eric Steven Raymond.** "The Linux Installation HOWTO". *The Linux Documentation Project.*

[Rad] **Jacek Radajewski; Douglas Eadline** (2002). "Beowulf: Installation and Administration". In: Kurt Swendson. *Beowulf HOWTO (tlpd).*
<http://www.sci.usq.edu.au/staff/jacek/beowulf>

[Red] Optimisation of Linux servers.
<http://people.redhat.com/alikins/system_tuning.html>

[Redb] System-config-cluster (FC).
<http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Cluster_Administration/index.htm>

[Redh] Red Hat Inc. "Red Hat Distribution".
<http://www.redhat.com>

[Rid] **Daniel Lopez Ridruejo** (2000). "The Linux Networking Overview". *The Linux Documentation Project.*

[Rus] **Rusty Russell.** "Linux IPCHAINS". *The Linux Documentation Project.*

[SM02] **Michael Schwartz and other** (2002). *Multitool Linux - Practical Uses for Open Source Software.* Addison Wesley.

[Sal94] **Peter H. Salus** (1994). "25th anniversary of UNIX" (no. 1, November). *Byte Spain.*

[Sam] Samba Project.
<http://samba.org>

[Sama] Samba HOWTO and Reference Guide (Chapter Domain Control).
<http://samba.org/samba/docs/man/Samba-HOWTO-Collection/samba-pdc.html>

[Samb] Samba Guide (Chapter Adding Domain member Servers and Clients).
<http://samba.org/samba/docs/man/Samba-Guide/unixclients.html>

[San] **Sans.** "Top20 vulnerabilities".
<http://www.sans.org/top20/>

[Sci] Scientific Linux.
<http://www.scientificlinux.org>

[Sec] **Andrés Seco** (2000). "Diald". *The Linux Documentation Project.*

[Sei02] **Kurt Seifried** (2002). "Securing Linux, Step by Step".
<http://seifried.org/security/os/linux/20020324-securing-linux-step-by-step.html>

[Skoa] **Miroslav Skoric.** "LILO mini-HOWTO". *The Linux Documentation Project.*

[Skob] **Miroslav Skoric.** "Linux+WindowsNT mini-HOWTO". *The Linux Documentation Project.*

[Sla] **Slashdot.** "Slashdot site".
<http://slashdot.org>

[Smb] Wikipedia entry for "Server Message Block".
<http://en.wikipedia.org/wiki/Server_Message_Block>

[Smi02] **Rod Smith** (2002). *Advanced Linux Networking.* Addison Wesley.

[Sno] **Snort.org**. *Snort.*
<http://www.snort.org>

[Sou] **Sourceforge.** "Sourceforge site".
<http://sourceforge.org>

[Squ] Squid proxy server.
<http://www.squid-cache.org/>

[Sta02] **Richard Stallman** (2002). "Discussion by Richard Stallman on relationship between GNU and Linux".
<http://www.gnu.org/gnu/linux-and-gnu.html>

[Stu] **Michael Stutz.** "The Linux Cookbook: Tips and Techniques for Everyday Use". *The Linux Documentation Project* (guías).

[Ste07] Steve French, Linux CIFS Client guide.
<http://us1.samba.org/samba/ftp/cifs-cvs/linux-cifs-client-guide.pdf>

[Stei] **Tony Steidler-Dennison** (2005). Your Linux Server and Network. Sams.

[Sub] Subversion.
<http://subversion.tigris.org>

[Subb] Control of versions with Subversion. Free Book.
<http://svnbook.red-bean.com/index.es.html>

*[Sun02] **Rahul Sundaram** (2002). "The dosemu HOWTO". *The Linux Documentation Project.*

[Sun] **Sun.** "Sun Grid Engine".
<http://www.sun.com/software/gridware/>

[Tan87] **Andrew Tanenbaum** (1987). *Operating system: Design and Implementation.* Prentice Hall.

[Tan06] **Andrew Tanenbaum; Albert S. Woodhull** (2006). *The Minix Book: Operating Systems Design and Implementation* (3rd ed.). Prentice Hall.

[Tkc] **Tkcvs** (2003). "Tkcvs interface for CVS".
<http://www.tkcvs.org>
<http://www.twobarleycorns.net/tkcvs.html>

[Tri] **Tripwire.com.** *Tripwire Web Site.*
<http://www.tripwire.com/>

[Tum02] **Enkh Tumenbayar** (2002). "Linux SMP HOWTO". *The Linux Documentation Project.*

[Ubn] Ubuntu Distribution.
<http://www.ubuntu.com>

[Uni] **Wisconsin University** (2003). *Condor Web Site.*
<http://www.cs.wisc.edu/condor>

[USA] **Dep. Justice USA.** "Division of the US Justice Department for cybercrime".
<http://www.usdoj.gov/criminal/cybercrime/>

[Vah96] **Uresh Vahalia** (1996). *UNIX Internals: The New Frontiers.* Prentice Hall.

[Vas] **Alavoor Vasudevan** (2000). "Modem-Dialup-NT". *The Linux Documentation Project.*

[Vasa] **Alavoor Vasudevan** (2003). "CVS-RCS (Source Code Control System)". *The Linux Documentation Project.*

[Vasb] **Alavoor Vasudevan.** "The Linux Kernel HOWTO". *The Linux Documentation Project.*

[Wm02] **Matt Welsh and others** (2002). *Running Linux 4th edition.* O'Reilly.

[War] **Ian Ward.** "Debian and Windows Shared Printing mini-HOWTO". *The Linux Documentation Project.*

[Web] **Webmin.***Tool for administrating Linux systems.*
<http://www.webmin.com/>

[Wil02] **Matthew D. Wilson** (2002). "VPN". *The Linux Documentation Project.*

[Win] Wine Project.
<http://www.winehq.com/

[Wir] WireShark.
<http://www.wireshark.org/download.html>

[Woo] **David Wood.** "SMB HOWTO". *The Linux Documentation Project.*

[Xin] Xinetd Web Site.
<http://www.xinetd.org/>

[Zan] **Renzo Zanelli.***Win95 + WinNT + Linux multiboot using LILOmini-HOWTO. The Linux Documentation Project.*

GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine- readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent.

An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard- conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or

XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language (here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History"). To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly

and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition.

Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine- readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material.

If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number.

Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit.

When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form.

Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright hold-

ers, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http:// www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number.

If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the

Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page: