The openMosix HOWTO

Live free() or die()

Kris Buytaert

<buytaert@x-tend.be>

and Others

Revision History

Revision v1.0.3	18 june 2004
Minor Fixes	
Revision v1.0.2	29 july 2003
RPM Build	
Revision v1.0.1	19 july 2003
Major updates	
Revision v1.0	09 july 2003
Minor updates	
Revision v1.0	11 may 2003
At last	
Revision v1.0 RC 1	07 may 2003
Major Cleaning	
Revision v0.95	04 april 2003
Replaced ClumpOS by PlumpOS	
Revision v0.94	25 february 2003
Patches by Mirko Caserta	
Revision v0.93	16 february 2003
Extra features and fixes	
Revision v0.92	21 january 2003
Revision v0.91	27 september 2002
Revision v0.90	03 september 2002
Revision v0.71	26 August 2002
Spleling Fexis	
Revision v0.70	22 August 2002
Stripped out empty parts, replaced Mosixview with openMosixView	
Revision v0.50	6 July 2002

First openMosix HOWTO	
Revision v0.20	5 July 2002
Latest Mosix HOWTO (for now)	
Revision v0.17	28 June 2002
Revision v0.15	13 March 2002
Revision v0.13	18 Feb 2002
Revision ALPHA 0.03	09 October 2001

"The best way to become acquainted with a subject is to write a book about it." (Benjamin Disraeli)

Table of Contents

Table of Contents

Table of Contents

Chapter 10. openMosixview

Table of Contents

Table of Contents

Chapter 1. Introduction

1.1. openMosix HOWTO

In the beginning there was Mosix, then came openMosix, in my opinion a more interesting project. Not only from a technical point of view but also due to the more correct license. I made the decision to focus this HOWTO on openMosix rather than on Mosix, mainly based on the fact that openMosix has a bigger userbase. (Moshe Bar states that about 97% of the old Mosix community has switched over to openMosix.) (20020705) Given the above, lots of information might be valuable to both users of Mosix and openMosix. I decided to split the HOWTO. The latest release of the Mosix HOWTO, containing info about both Mosix and OpenMosix will be 0.20 My intention is to focus on the openMosix HOWTO, however not neglecting the Mosix users. More info on *http://howto.ipng.be/Mosix−HOWTO/*

1.2. Introduction

This document gives a brief description of openMosix, a software package that turns a network of GNU/Linux computers into a computer cluster. Along the way, some background to parallel processing is given, as well as a brief introduction to programs that make special use of openMosix's capabilities. The HOWTO expands on the documentation as it provides more background information and discusses the quirks of various distributions.

Since the creation of this HOWTO some people of the Mosix team created openMosix (more info later), initially both openMosix and Mosix were discussed in this HOWTO. Although lots of information might be valuable to both users of Mosix and openMosix. I decided to split the HOWTO. The latest relase of the Mosix HOWTO, containing info about both Mosix and OpenMosix will be 0.20 and can be found on *http://howto.ipng.be/Mosix−HOWTO/Mosix−HOWTO/*

Kris Buytaert got involved in this piece of work when Scot Stevenson was looking for somebody to take over the Job: this was during February 2002. While initially we discussed both Mosix and openMosix, this version of the HOWTO now mainly focuses on openMosix. Please note that the document often still mentions Mosix where it should read openMosix.

You will notice that some of the headings are not as serious as they should be. Scot had planned to write the HOWTO in a slightly lighter style, as the world (and even the part of the world with a burping penguin as a mascot) is full of technical literature that is deadly. Therefore some parts still have these comments.

1.3. Disclaimer

Use the information in this document at your own risk. I disavow potential liability for the contents of this document. Use of these concepts, examples, and/or other content of this document is entirely at your own risk.

All copyrights are owned by their respective owners, unless specified otherwise. Use of a term in this document should not be regarded affecting the validity of any trademark or service mark. openMosix is Copyright (c) by Moshe Bar. Mosix is Copyright (c) by Amnon Barak. Linux is a Registered Trademark of Linus Torvalds. openMosix is licensed under version 2 of the GNU General Public License as published by the Free Software Foundation.

Naming of particular products or brands should not be seen as endorsements.

You are strongly recommended to take a backup of your system before major installation and backups at regular intervals.

1.4. Distribution policy

1.5. New versions of this document

Official New versions of this document can be found on the web pages of *the Linux Documentation Project* Drafts and Beta versions will be available on *howto.ipng.be* in the appropriate sub folder. Changes to this document will usually be discussed on the openMosix Mailing Lists. See the *openMosix* for details.

1.6. Feedback

Currently this HOWTO is being maintained by Kris Buytaert. Please do send remarks and updates for the howto to him.

If you have a technical question about openMosix/Mosix itself, please post them on the more appropriate mailing list.

Chapter 2. So what is openMosix Anyway ?

2.1. A very, very brief introduction to clustering

Most of the time, your computer is bored. Start a program like xload or top that monitors your system use, and you will probably find that your processor load is not even hitting the 1.0 mark. If you have two or more computers, chances are that at any given time, at least one of them is doing nothing. Unfortunately, when you really do need CPU power − during a C++ compile, or encoding Ogg Vorbis music files − you need a lot of it at once. The idea behind clustering is to spread these loads among all available computers, using the resources that are free on other machines.

The basic unit of a cluster is a single computer, also called a "node". Clusters can grow in size − they "scale" − by adding more machines. A cluster as a whole will be more powerful the faster the individual computers and the faster their connection speeds are. In addition, the operating system of the cluster must make the best use of the available hardware in response to changing conditions. This becomes more of a challenge if the cluster is composed of different hardware types (a "heterogeneous" cluster), if the configuration of the cluster changes unpredictably (machines joining and leaving the cluster), and the loads cannot be predicted ahead of time.

2.1.1. A very, very brief introduction to clustering

2.1.1.1. HPC vs Fail−over vs Load−balancing

Basically there are 3 types of clusters, Fail−over, Load−balancing and HIGH Performance Computing, The most deployed ones are probably the Failover cluster and the Load−balancing Cluster.

- *Fail−over Clusters* consist of 2 or more network connected computers with a separate heartbeat connection between the 2 hosts. The Heartbeat connection between the 2 machines is being used to monitor whether all the services are still in use: as soon as a service on one machine breaks down the other machines try to take over.
- With load−balancing clusters the concept is that when a request for say a web−server comes in, the cluster checks which machine is the least busy and then sends the request to that machine. Actually most of the times a Load−balancing cluster is also a Fail−over cluster but with the extra load balancing functionality and often with more nodes.
- The last variation of clustering is the High Performance Computing Cluster: the machines are being configured specially to give data centers that require extreme performance what they need. Beowulfs have been developed especially to give research facilities the computing speed they need. These kind of clusters also have some load−balancing features; they try to spread different processes to more machines in order to gain performance. But what it mainly comes down to in this situation is that a process is being parallelized and that routines that can be ran separately will be spread on different machines instead of having to wait till they get done one after another.

Most common known examples of loadbalancing and failover clusters are webfarms, databases or firewalls. People want to have a 99,99999% uptime for their services, the internet is open 24/24 7/7/ 365/365 not unlike in the old days when you could shut down your server when the office closed.

People that are in need of cpu cycles often can afford to schedule downtime for their environments, as long as they can use the maximum power of their machines when they need it.

2.1.1.2. Supercomputers vs. clusters

Traditionally Supercomputers have only been built by a selected number of vendors: a company or organization that required the performance of such a machine had to have a huge budget available for its Supercomputer. Lots of universities could not afford the costs of a Supercomputer by themselves, therefore other alternatives were being researched by them. The concept of a cluster was born when people first tried to spread different jobs over more computers and then gather back the data those jobs produced. With cheaper and more common hardware available to everybody, results similar to real Supercomputers were only to be dreamed of during the first years, but as the PC platform developed further, the performance gap between a Supercomputer and a cluster of multiple personal computers became smaller.

2.1.1.3. Cluster models [(N)UMA, PVM/MPI]

There are different ways of doing parallel processing: (N)UMA, DSM, PVM and MPI are all different kinds of Parallel Processing schemes. Some of them are implemented in hardware, others in software, others in both.

(N)UMA ((Non−)Uniform Memory Access), machines for example have shared access to the memory where they can execute their code. In the Linux kernel there is a NUMA implementation that varies the memory access times for different regions of memory. It then is the kernel's task to use the memory that is the closest to the CPU it is using.

DSM aka Distributed Shared memory, has been implemented in both software and hardware , the concept is to provide an abstraction layer for physically distributed memory.

PVM and MPI are the tools that are most commonly being used when people talk about GNU/Linux based Beowulfs.

MPI stands for Message Passing Interface. It is the open standard specification for message passing libraries. MPICH is one of the most used implementations of MPI. Next to MPICH you also can find LAM, another implementation of MPI based on the free reference implementation of the libraries.

PVM or Parallel Virtual Machine is another cousin of MPI that is also quite often being used as a tool to create a Beowulf. PVM lives in user space so no special kernel modifications are required: basically each user with enough rights can run PVM.

2.1.1.4. openMosix's role

The openMosix software package turns networked computers running GNU/Linux into a cluster. It automatically balances the load between different nodes of the cluster, and nodes can join or leave the running cluster without disruption of the service. The load is spread out among nodes according to their connection and CPU speeds.

Since openMosix is part of the kernel and maintains full compatibility with Linux, a user's programs, files, and other resources will all work as before without any further changes. The casual user will not notice the difference between a Linux and an openMosix system. To her, the whole cluster will function as one (fast) GNU/Linux system.

openMosix is a Linux−kernel patch which provides full compatibility with standard Linux for IA32−compatible platforms. The internal load−balancing algorithm transparently migrates processes to other cluster members. The advantage is a better load−sharing between the nodes. The cluster itself tries to optimize utilization at any time (of course the sysadmin can affect the automatic load−balancing by manual configuration during runtime).

This transparent process−migration feature makes the whole cluster look like a BIG SMP−system with as many processors as available cluster−nodes (of course multiplied with X for X−processor systems such as dual/quad systems and so on). openMosix also provides a powerful optimized File System (oMFS) for HPC−applications, which unlike NFS provides cache, time stamp and link consistency.

2.2. The story so far

2.2.1. Historical Development

Rumours say that Mosix comes from Moshe Unix. Initially Mosix started out as an application running on BSD/OS 3.0.

```
Announcing MO6 for BSD/OS 3.0
Oren Laadan (orenl@cs.huji.ac.il)
Tue, 9 Sep 1997 19:50:12 +0300 (IDT)

Hi:

We are pleased to announce the availability of MO6 Version 3.0
Release 1.04 (beta−4) − compatible with BSD/OS 3.0, patch level
K300−001 through M300−029.

MO6 is a 6 processor version of the MOSIX multicomputer enhancements
of BSD/OS for a PC Cluster. If you have 2 to 6 PC's connected by a
LAN, you can experience truly multi−computing environment by using
the MO6 enhancements.

The MO6 Distribution
−−−−−−−−−−−−−−−−−−−−
MO6 is available either in "source" or "binary" distribution. It is
installed as a patch to BSD/OS, using an interactive installation
script.

MO6 is available at http://www.cnds.jhu.edu/mirrors/mosix/
or at our site: http://www.cs.huji.ac.il/mosix/

Main highlights of the current release:
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
− Memory ushering (depletion prevention) by process migration.
− Improved installation procedure.
− Enhanced migration control.
− Improved administration tools.
− More user utilities.
− More documentation and new man pages.
− Dynamic configurations.

Please send feedback and comments to mosix@cs.huji.ac.il.
−−−−−−−−−−−−−−−−−−−
```

GNU/Linux was chosen as a development platform for the 7th incarnation in 1999. Early 1999 Mosix M06 Beta was released for Linux 2.2.1 At the end of 2001 and early 2002 openMosix, the open version of Mosix was born (more in the next paragraph).

2.2.2. openMosix

openMosix is in addition to whatever you find at mosix.org and in full appreciation and respect for Prof. Barak's leadership in the outstanding Mosix project.

Moshe Bar has been involved for a number of years with the Mosix project (www.mosix.com) and was co−project manager of the Mosix project and general manager of the commercial Mosix company.

After a difference of opinions on the commercial future of Mosix, he has started a new clustering company − Qlusters, Inc. − and Prof. Barak has decided not to participate for the moment in this venture (although he did seriously consider joining) and held long running negotiations with investors. It appears that Mosix is not any longer supported openly as a GPL project. Because there is a significant user base out there (about 1000 installations world−wide), Moshe Bar has decided to continue the development and support of the Mosix project under a new name: openMosix and under the full GPL2 license. Whatever code in openMosix comes from the old Mosix project is Copyright 2002 by Amnon Barak. All the new code is Copyright 2002 by Moshe Bar.

There could (and will) be significant changes in the architecture of the future openMosix versions. New concepts about auto−configuration, node−discovery and new user−land tools are being discussed in the openMosix mailing lists. Most of these new functionalities are already implemented while some of them, such as DSM (Distributed Shared Memory) are still being worked on at the moment I write this (march 2003).

To approach standardization and future compatibility the proc−interface has changed from /proc/mosix to /proc/hpc and the /etc/mosix.map was changed to /etc/hpc.map. More recently the standard for the config file has been set to be located in /etc/openmosix.map (this is in fact the first config file the /etc/init.d/openmosix script will look for). Adapted command−line user−space tools for openMosix are already available on the web−page of the project.

The openmosix.map config file can be replaced with a node−auto−discovery system which is called omdiscd (openMosix auto DISCovery Daemon) about which we will discuss later.

openMosix is supported by various competent people (see openmosix.sourceforge.net) working together around the world. The main goal of the project is to create a standardized clustering−environment for all kinds of HPC−applications.

openMosix has also a project web−page at *http://openMosix.sourceforge.net* with a CVS tree and mailing−lists for developers as well as users.

2.2.3. Current state

Like most active Open Source programs, openMosix's rate of change tends to outstrip the followers' ability to keep the documentation up to date.

As I write this part in February 2003 openMosix 2.4.20 is available and openMosix Userland Tools v0.2.4 are available, including the new autodiscovery tools.

For a more recent state of development please take a look at the *openMosix website*

2.2.4. Which applications work

It is almost impossible to give a list off all the applications that work with openMosix. The community however tries to keep track of the applications that *migrate* and the ones who *don't*.

2.3. openMosix in action: An example

openMosix clusters can take various forms. To demonstrate this, let's assume you are a student and share a dorm room with a rich computer science guy, with whom you have linked computers to form an openMosix cluster. Let's also assume you are currently converting music files from your CDs to Ogg Vorbis for your private use, which is legal in your country. Your roommate is working on a project in C++ that he says will bring World Peace. However, at just this moment he is in the bathroom doing unspeakable things, and his computer is idle.

So when you start a program like bladeenc to convert Bach's .... from .wav to .ogg format, the openMosix routines on your machine compare the load on both nodes and decide that things will go faster if that process is sent from your Pentium−233 to his Athlon XP. This happens automatically: you just type or click your commands as you would if you were on a standalone machine. All you notice is that when you start two more coding runs, things go a lot faster, and the response time doesn't go down.

Now while you're still typing ...., your roommate comes back, mumbling something about red chili peppers in cafeteria food. He resumes his tests, using a program called 'pmake', a version of 'make' optimized for parallel execution. Whatever he's doing, it uses up so much CPU time that openMosix even starts to send subprocesses to your machine to balance the load.

This setup is called *single−pool*: all computers are used as a single cluster. The advantage/disadvantage of this is that your computer is part of the pool: your stuff will run on other computers, but their stuff will run on yours too.

2.4. Components

2.4.1. Process migration

With openMosix you can start a process on one machine and find out it actually runs on another machine in the cluster. Each process has its own Unique Home Node (UHN) where it gets created.

Migration means that a process is splitted in 2 parts, a user part and a system part. The user part will be moved to a remote node while the system part will stay on the UHN. This system−part is sometimes called the deputy process: this process takes care of resolving most of the system calls.

openMosix takes care of the communication between these 2 processes.

2.4.2. The openMosix File System (oMFS)

oMFS is a feature of openMosix which allows you to access remote filesystems in a cluster as if they were locally mounted. The filesystems of your other nodes can be mounted on /mfs and you will, for instance, find the files in /home on node 3 on each machine in /mfs/3/home.

2.4.3. Direct File System Access (DFSA)

Both Mosix and openMosix provide a cluster−wide file−system (MFS) with the DFSA−option (Direct File−System Access). It provides access to all local and remote file−systems of the nodes in a Mosix or openMosix cluster.

2.5. openMosix Test Drive

In support of openMosix, Major Chai Mee Joon is giving OM users a free trial account to his online openMosix cluster service, which users can use to test and experiment openMosix with.

The availability of this online openMosix cluster service will help both new users overcome the initial openMosix configuration issues, and also provides higher computing power to openMosix users who are developing or porting their applications.

Please send an email to <`om@majorlinux.com`> for a trial account.

2.6. Pros of openMosix

Table 2−1. Pros of openMosix

No extra packages are required.
No code changes to your application are required.
Simple to install/configure.
On a Red−Hat based system/distro, installing openMosix is as simple as typing: # rpm −Uvh openMosix*.rpm
DSM is being released soon (late march 2003).
Well integrated with openAFS.
Port to IA−64 as well as AMD−64 is underway.
oMFS has been improved much since plain MFS.
It is a clustering platform with more than 10 products based on it: openMosixView, openMosixWebView, openMosixApplet, RxLinux, PlumpOS, K12LTSP, LTSP and many others.
openMosix is a product developed by the users themselves so it's more close to the user by definition.
Node autodiscovery/fail−over daemon already implemented in the user land tools via multicast messaging.
Aliases for hosts with multiple interfaces.
Basic routing available (in the rare case where true multicast routing is undesirable).
Cluster Mask allows to specify to which nodes a given process can migrate.

2.7. Cons of openMosix

Table 2−2. Cons of openMosix

Kernel dependent.

Shared memory issues (an alpha release of DSM should be available as of late march 2003).
There are issues with Multiple Threads not gaining performance.
You won't gain performance when running one single process such as your web browser on an openMosix Cluster: the process won't spread itself over the cluster. Except of course your process will migrate to a more performant machine.

II. Installing openMosix

Table of Contents

Chapter 3. Requirements and Planning

3.1. Hardware requirements

Installing a basic cluster requires at least 2 network connected machines, either using a cross−cable between the two network cards or using a switch or hub (a switch is much better than a hub though and only costs a few bucks more). Of course the faster your network−cards the easier you will get better performance for your cluster.

These days Fast Ethernet (100 Mbps) is standard; putting multiple ports in a machine isn't that difficult, but make sure to connect them through other physical networks in order to gain the speed you want. Gigabit Ethernet is getting cheaper every day now but I suggest that you don't rush to the shop spending your money before you have actually tested your setup with multiple 100Mbit cards and noticed that you really do need the extra network capacity. Next to putting a Gigabit card you might also want to try bonding different 100Mbit cards together. An even cheaper alternative can be found in Firewire, as discussed in *this paper*

3.2. Hardware Setup Guidelines

Setting up a big cluster requires some thinking to be done: where are you going to put the machines? Not under a table somewhere or in the middle of your office I hope! It's ok if you just want to do some small tests, but if you are planning to deploy a N node cluster you will have to make sure that the environment that will hold these machines is capable of doing so.

I'm talking about preparing one or more 19" racks to host the machines, configuring the appropriate network topology, either straight, single connected or even a 1 to 1 cross connected network between all your nodes. You will also need to make sure that there is enough power to support such a range of machines, that your air−conditioning system supports the load and that in case of power−failure your UPS can cleanly shut down all the required systems. You might want to invest in a KVM (Keyboard, Video, Mouse) Switch in order to facility access to the machines' consoles.

But even if you don't have the number of nodes that justifies such an investment, make sure that you can always easily access the different nodes, you never know when you have to replace a CPU fan or an hard−disk of a machine in trouble. If that means that you have to unload a stack of machines to reach the bottom one, hence shutting down your cluster, you are in trouble.

3.3. Software requirements

The systems we plan to use will need a basic Linux installation of your choice: Red Hat, SuSe, Debian, Gentoo or any another distribution: it doesn't really matter which one. What does matter is that the kernel is at least on 2.4 level, and that your network−cards are configured correctly; next to that you'll need a healthy space of swap.

3.4. Planning your Cluster

When it comes to configuring openMosix Clusters with a pool of servers and a set of (personal) workstations, you have different options that will have their advantages and disadvantages.

- In a *Single−pool* configuration all the servers and workstations are used as a single cluster: each machine is a part of the cluster and can migrate processes to each other existing node. This of course makes your workstation a part of the pool.
- In an environment that is called a *Server−pool*, servers are a part of the cluster while workstations aren't part of it, they don't even have openMosix kernel. If you want to run applications on the cluster you will need to specifically log on to these servers. However your workstation will also stay clean and no remote processes will migrate to it.
- A third alternative is called an *Adaptive−pool* configuration: here servers are shared while workstations join or leave the cluster. Imagine your workstation being used during daytime by yourself but, as soon as you log out in the evening, a script tells the workstation to join the cluster and start crunching numbers. This way your machine is being used while you don't need it. If you need the resources of the machine again just run the openmosix stop script and your processes will stay away from the cluster and vice−versa.

  Practically this means that you will change the role of your machine by using mosctl.

3.5. Classrooms

Although it might seem a good idea to convert your classroom into an openMosix cluster at night, you'll have to consider training your end users not to pull the power switch of those machines when they want to use them again. More recent machines support automatic shutdowns when hitting the power button, but with older machines you might loose some data when this actually happens.

Chapter 4. Distribution specific installations

4.1. Installing openMosix

This chapter deals with installing openMosix on different distributions. It won't be an exhaustive list of all the possible combinations. However throughout the chapter you should find enough information on installing openMosix in your environment.

Techniques for installing multiple machines with openMosix will be discussed in one of the next chapters.

4.2. Before getting openMosix

First of all, you must understand that openMosix is made up of a kernel patch and some user−space tools. The kernel patch is needed to make the kernel capable of talking to other openMosix−enabled machines on the network. If you download openMosix as a binary package (such as an rpm file), you don't even need to take care about the kernel patch because the kernel has been patched and compiled with the most common default options and modules for you.

The user−space tools are needed in order to make an effective use of an openMosix−enabled kernel. They are needed to start/stop the migration daemon, the openMosix File System, to migrate jobs to certain nodes and other tasks which are usually accomplished with the help our good old friend: the command line interface. About binary packages: the same as in the kernel patch goes for the user−space tools: if you install an rpm you don't need to care about compiling them or configuring anything; just let them install and run. That's all. Really :)

Once you get to the download page (which we'll talk about in a second), you'll need to get two distinct parts: the kernel and the user−space tools. You can either download two binary packages or get the kernel patch plus the user−space tools' sources. The kernel patch is usually named after this scheme: openMosix−x.y.z−w where x.y.z is the version of the vanilla Linux Kernel against which the patch should be applied and w is the patch revision for that particular kernel release. For the precompiled kernel binaries, please refer to the README−openMosix−kernel.txt file you'll find in the download page. This file also contains updated info about manually compiling a kernel.

About the user−space tools: you'll find those in a package named openmosix−tools. We use the terms user−space tools, userspace−tools and openmosix−tools interchangeably. Updated info about precompiled binaries and manually compiling the tools are also provided in the README−openmosix−tools.txt file. Please note that since version 0.3 of the openmosix−tools, the openmosix.map file is deprecated and the use of the autodiscovery daemon is highly encouraged since it tends to make your life easier.

4.3. Getting openMosix

You can download the latest versions of openMosix from *http://sourceforge.net/project/showfiles.php?group_id=46729*. You can either choose the binary (even in rpm) compiled for UP or SMP or download the source code. You will need both the kernel patch or binaries and the userland tools. Alternatively you can get the CVS version:

```
cvs −d:pserver:anonymous@cvs.openmosix.sourceforge.net:/cvsroot/openmosix login
cvs −z3 −d:pserver:anonymous@cvs.openmosix.sourceforge.net:/cvsroot/openmosix co linux−openmosix
cvs −z3 −d:pserver:anonymous@cvs.openmosix.sourceforge.net:/cvsroot/openmosix co userspace−tools
```

At the password prompt, just type enter since you're doing an anonymous login. Please take care that CVS trees DO BREAK now and then and that it might not be the easiest way to install openMosix ;−)

4.4. openMosix General Instructions

4.4.1. Kernel Compilation

Always use pure vanilla kernel−sources from *http://www.kernel.org/* to compile an openMosix kernel! Please be kind enough to download the kernel using a mirror near to you and always try and download patches to the latest kernel sources you do have instead of downloading the whole thing. This is going to be much appreciated by the Linux community and will greatly increase your geeky Karma ;−) Be sure to use the right openMosix patch depending on the kernel−version. At the moment I write this, the latest 2.4 kernel is 2.4.20 so you should download the openMosix−2.4.20−x.gz patch, where the "x" stands for the patch revision (ie: the greater the revision number, the most recent it is). Do not use the kernel that comes with any Linux−distribution: it won't work. These kernel sources get heavily patched by the distribution−makers so, applying the openMosix patch to such a kernel is going to fail for sure! Been there, done that: trust me ;−)

Download the actual version of the openMosix patch and move it in your kernel−source directory (e.g. /usr/src/linux−2.4.20). If your kernel−source directory is other than "/usr/src/linux−[version_number]" at least the creation of a symbolic link to "/usr/src/linux−[version_number]" is required. Supposing you're the root user and you've downloaded the gzipped patch file in your home directory, apply the patch using (guess what?) the patch utility:

```
mv /root/openMosix−2.4.20−2.gz /usr/src/linux-2.4.20
cd /usr/src/linux-2.4.20
zcat openMosix−2.4.20−2.gz | patch −Np1
```

In the rare case you don't have "zcat" on your system, do:

```
mv /root/openMosix−2.4.20−2.gz /usr/src/linux-2.4.20
cd /usr/src/linux-2.4.20
gunzip openMosix−2.4.20−2.gz
cat openMosix−2.4.20−2 | patch −Np1
```

If the even more weird case you don't have a "cat" on your system (!), do:

```
mv /root/openMosix−2.4.20−2.gz /usr/src/linux-2.4.20
cd /usr/src/linux-2.4.20
gunzip openMosix−2.4.20−2.gz
patch −Np1 < openMosix−2.4.20−2
```

The "patch" command should now display a list of patched files from the kernel−sources. If you feel adventurous enough, enable the openMosix related options in the kernel−configuration file, e.g.

```
...
CONFIG_MOSIX=y
# CONFIG_MOSIX_TOPOLOGY is not set
CONFIG_MOSIX_UDB=y
# CONFIG_MOSIX_DEBUG is not set
# CONFIG_MOSIX_CHEAT_MIGSELF is not set
CONFIG_MOSIX_WEEEEEEEEE=y
CONFIG_MOSIX_DIAG=y
CONFIG_MOSIX_SECUREPORTS=y
CONFIG_MOSIX_DISCLOSURE=3
CONFIG_QKERNEL_EXT=y
```

```
CONFIG_MOSIX_DFSA=y
CONFIG_MOSIX_FS=y
CONFIG_MOSIX_PIPE_EXCEPTIONS=y
CONFIG_QOS_JID=y
...
```

However, it's going to be pretty much easier if you configure the above options using one of the Linux−kernel configuration tools:

```
make config | menuconfig | xconfig
```

The above means you have to choose one of "config", "menuconfig", and "xconfig". It's a matter of taste. By the way, "config" is going to work on any system; "menuconfig" needs the curses libraries installed while "xconfig" needs an installed X−window environment plus the TCL/TK libraries and interpreters.

Now compile it with:

```
make dep bzImage modules modules_install
```

After compilation install the new kernel with the openMosix options within you boot−loader; e.g. insert an entry for the new kernel in /etc/lilo.conf and run lilo after that.

Reboot and your openMosix−cluster−node is up!

4.4.2. Syntax of the /etc/openmosix.map file

Before starting openMosix, there has to be an /etc/openmosix.map configuration file which must be the same on each node.

The standard is now /etc/openmosix.map, /etc/mosix.map and /etc/hpc.map are old standards, but the CVS−version of the tools is backwards compatible and looks for /etc/openmosix.map, /etc/mosix.map and /etc/hpc.map (in that order).

The openmosix.map file contains three space separated fields:

```
openMosix−Node_ID               IP−Address(or hostname)          Range−size
```

An example openmosix.map file could look like this:

```
1       node1   1
2       node2   1
3       node3   1
4       node4   1
```

or

```
1       192.168.1.1     1
2       192.168.1.2     1
3       192.168.1.3     1
4       192.168.1.4     1
```

or with the help of the range−size both of the above examples equal to:

```
1       192.168.1.1     4
```

openMosix "counts−up" the last byte of the ip−address of the node according to its openMosix−Node_ID. Of course, if you use a range−size greater than 1 you have to use ip−addresses instead of hostnames.

If a node has more than one network−interface it can be configured with the ALIAS option in the range−size field (which equals to setting the range−size to 0) e.g.

```
1       192.168.1.1     1
2       192.168.1.2     1
3       192.168.1.3     1
4       192.168.1.4     1
4       192.168.10.10   ALIAS
```

Here the node with the openMosix−Node_ID 4 has two network−interfaces (192.168.1.4 + 192.168.10.10) which are both visible to openMosix.

Always be sure to run the same openMosix version AND configuration on each of your Cluster's nodes!

Start openMosix with the "setpe" utility on each node :

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

Execute this command (which will be described later on in this HOWTO) on every node in your openMosix cluster.

Alternatively, you can grab the "openmosix" script which can be found in the scripts directory of the userspace−tools, copy it to the /etc/init.d directory, chmod 0755 it, then use the following commands as root:

```
/etc/init.d/openmosix stop
/etc/init.d/openmosix start
/etc/init.d/openmosix restart
```

Installation is finished now: the cluster is up and running :)

4.4.3. oMFS

First of all, the CONFIG_MOSIX_FS option in the kernel configuration has to be enabled. If the current kernel was compiled without this option, then recompilation with this option enabled is required.

Also the UIDs (User IDs) and GIDs (Group IDs) on each of the clusters' nodes file−systems must be the same. You might want to accomplish this using openldap. The CONFIG_MOSIX_DFSA option in the kernel is optional but of course required if DFSA should be used. To mount oMFS on the cluster there has to be an additional fstab−entry on each node's /etc/fstab.

in order to have DFSA enabled:

```
mfs_mnt         /mfs            mfs     dfsa=1         0 0
```

in order to have DFSA disabled:

```
mfs_mnt          /mfs           mfs     dfsa=0         0 0
```

the syntax of this fstab−entry is:

```
[device_name]           [mount_point]   mfs     defaults        0 0
```

After mounting the /mfs mount−point on each node, each node's file−system is going to be accessible through the /mfs/[openMosix−Node_ID]/ directories.

With the help of some symbolic links all cluster−nodes can access the same data e.g. /work on node1

```
on node2 :      ln −s /mfs/1/work /work
on node3 :      ln −s /mfs/1/work /work
on node3 :      ln −s /mfs/1/work /work
...
```

Now every node can read+write from and to /work !

The following special files are excluded from the oMFS:

- the /proc directory
- special files which are not regular−files, directories or symbolic links (e.g. /dev/hda1)

Creating links like:

```
ln −s /mfs/1/mfs/1/usr
```

or

```
ln −s /mfs/1/mfs/3/usr
```

is invalid.

The following system calls are supported without sending the migrated process (which executes this call on its home (remote) node) going back to its home node:

read, readv, write, writev, readahead, lseek, llseek, open, creat, close, dup, dup2, fcntl/fcntl64, getdents, getdents64, old_readdir, fsync, fdatasync, chdir, fchdir, getcwd, stat, stat64, newstat, lstat, lstat64, newlstat, fstat, fstat64, newfstat, access, truncate, truncate64, ftruncate, ftruncate64, chmod, chown, chown16, lchown, lchown16, fchmod, fchown, fchown16, utime, utimes, symlink, readlink, mkdir, rmdir, link, unlink, rename

Here are situations when system calls on DFSA mounted file−systems may not work:

- different mfs/dfsa configuration on the cluster−nodes
- dup2 if the second file−pointer is non−DFSA
- chdir/fchdir if the parent dir is non−DFSA
- pathnames that leave the DFSA−filesystem
- when the process which executes the system−call is being traced
- if there are pending requests for the process which executes the system−call

Next to the /mfs/1/ /mfs/2/ and so on files you will find some other directories as well.

Table 4−1. Other Directories

/mfs/here	The current node where your process runs
/mfs/home	Your home node
/mfs/magic	The current node when used by the "creat" system call (or an "open" with the "O_CREAT" option) − otherwise, the last node on which an oMFS magical file was successfully created (this is very useful for creating temporary−files, then immediately unlinking them)
/mfs/lastexec	The node on which the process last issued a successful "execve" system−call.
/mfs/selected	

	The node you selected by either your process itself or one of its ancestor's (before forking this process), writing a number into "/proc/self/selected".

Note that these magic files are all ``per process". That is their content is dependent upon which process opens them.

A last not about openMFS is that there are versions around that return faultive results when you run "df" on those filesystems. Don't be surpised if you suddenlty have about 1.3 TB available on those systems.

4.5. Red Hat and openMosix

If you are running a RedHat 7.2, 7.3 or 8.0 version, this is probably the easiest *Mosix install you have ever done. Choose the appropriate openMosix RPMs from sourceforge. They have precompiled kernels (as I write this 2.4.20) that work seamlessly: I have tested them on several machines including Laptops with PCMCIA cards and Servers with SCSI disks. If you are a grub user, the kernel rpm even modifies your grub.conf. So all you have to do is install 2 RPMs:

```
rpm −Uvh openmosix−kernel−2.4.20−openmosix2.i686.rpm openmosix−tools−0.2.4−1.i386.rpm
```

and edit your /etc/openmosix.map if you don't wish to use the autodiscovery daemon (omdiscd). Since this seems to be a problem for lots of people, let's go with another example. Say you have 3 machines: 192.168.10.220, 192.168.10.78 and 192.168.10.84. Your openmosix.map will look like this.

```
[root@oscar0 root]# more /etc/openmosix.map
# openMosix CONFIGURATION
# ===================
#
# Each line should contain 3 fields, mapping IP addresses to openMosix node−numbers:
# 1) first openMosix node−number in range.
# 2) IP address of the above node (or node−name from /etc/hosts).
# 3) number of nodes in this range.
#
# Example: 10 machines with IP 192.168.1.50 − 192.168.1.59
# 1        192.168.1.50     10
#
# openMosix−#  IP  number−of−nodes
# ============================
1 192.168.10.220 1
2 192.168.10.78  1
3 192.168.10.84  1
```

Now by rebooting the different machines with the newly installed kernel you will get one step closer to having a working cluster.

Most RedHat installations have one extra thing to fix. You often get the following error:

```
[root@inspon root]# /etc/init.d/openmosix start
Initializing openMosix...
setpe: the supplied table is well−formatted,
but my IP address (127.0.0.1) is not there!
```

This means that your hostname is not listed in /etc/hosts with the same ip as in your openmosix.map. You might have a machine called omosix1.localhost.org in your hostfile listed as

```
127.0.0.1       omosix1.localhost.org localhost
```

If you modify your /etc/hosts to look like below, openMosix will have less troubles starting up.

```
192.168.10.78   omosix1.localhost.org
127.0.0.1       localhost
```

```
[root@inspon root]# /etc/init.d/openmosix start
Initializing openMosix...
[root@inspon root]# /etc/init.d/openmosix status
This is openMosix node #2
Network protocol: 2 (AF_INET)
openMosix range     1−1     begins at 192.168.10.220
openMosix range     2−2     begins at inspon.localhost.be
openMosix range     3−3     begins at 192.168.10.84
Total configured: 3
```

If you would like to use more bleeding edge patches, you can always opt for the src rpm and run rpmbuild −−rebuild on it. This will install the source for you and create an initial config file. From there you can go further applying patches to openMosix

A tutorial on how to build your own openMosix RPM's can be found in the Appendixes.

As new RedHat versions come out, they might be supported out of the box so, feel free to drop the author a note and help him keeping this information updated.

4.6. Suse and openMosix

Although the RPMs are being built on a RedHat based environment, you can use most of them on other RPM based systems.

Suse however has /sbin/mk_initrd as a link to /sbin/mkinitrd, which makes rpms before release 20−2 fail. Newer version should have a fix for this.

4.7. Debian and openMosix

Installing openMosix ``the Debian way'' can be easily done as described below.

The first step consists in downloading the packages from the net. I had to use a 2.4.19 kernel since the openMosix patches package is not yet available for 2.4.20 at the moment I write this. Since we are using a Debian setup we needed: *http://packages.debian.org/unstable/net/openmosix.html*, *http://packages.debian.org/unstable/net/kernel−patch−openmosix.html*, *http://packages.debian.org/unstable/misc/kernel−package.html*, *http://packages.debian.org/unstable/devel/kernel−source−2.4.19.html*. You can also apt−get install them ;).

The next part is making the kernel openMosix capable.

Basically, the procedure to follow is:

```
cd /usr/src
apt−get install kernel−source−2.4.19 kernel−package \
        openmosix kernel−patch−openmosix
tar vxjf kernel−source−2.4.19.tar.bz2
ln −s /usr/src/kernel−source−2.4.19 /usr/src/linux
cd /usr/src/linux
../kernel−patches/i386/apply/openmosix
```

```
make menuconfig
make−kpkg kernel_image modules_image
cd ..
dpkg −i kernel−image−*−openmosix−*.deb
```

You now will need to edit your /etc/openmosix.map. Please follow the instructions given in the ``Syntax of /etc/openmosix.map'' part of this HOWTO.

After rebooting with this kernel and a configured /etc/openmosix.map, you should then have a cluster of openMosix machines that talk to each−other and that do migration of processes.

You can test that by running the following small script:

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

a couple of times, and monitor its behaviour with "mosmon" where you will see that it spreads the load between the different nodes.

We also setup openMosixView on the Debian machine:

```
apt−get install openmosixview
```

In order to be able to actually use openMosixView you will need to run it from a user who can log in to the different nodes as root. We suggest you set this up using ssh. Please note that there is a difference between the ssh and ssh2 implementations. If you do have an identity.pub file, ssh will check authorized_keys, while if you do have an id_dsa.pub you will need authorized_keys2!

openMosixView gives you a nice interface that shows the load of different machines and gives you the possibility to migrate processes manually.

A detailed discussion of openMosixView can be found elsewhere in this document.

4.8. openMosix and Gentoo

First Install Gentoo Linux

Then, install openMosix: type "emerge sys−apps/openmosix−user", which will install an openMosix kernel source tree in /usr/src/linux along with the openMosix userland tools.

Michael Imhof, aka tantive, keeps Gentoo current for the latest openMosix version.

Daniel Robbins, the President/CEO of Gentoo Technologies, Inc. and the creator of Gentoo Linux, wrote the artitles we use as our Introduction to openMosix Clusters.

4.9. Other distributions

Based on the explanations above you should be able to install openMosix on most other Linux platforms.

Chapter 5. Autodiscovery

5.1. Easy Configuration

The auto−discovery daemon (omdiscd) provides a way to automatically configure an openMosix cluster hence eliminating the need of a /etc/mosix.map or similar manual configurations. Auto−discovery uses multicast packages to notify other nodes that it is an openMosix node. This way adding an extra node to your mosix cluster means that you just have to start the omdiscd on your machine and it will join the cluster.

However there are some small requirements, Like with any openMosix cluster , you need to have networking configured correctly. mainly the routing. Without a default route, you must specify an interface to omdiscd with the −i option. Otherwise omdiscd will exit with an error like.

```
Aug 31 20:41:49 localhost omdiscd[1290]: Unable to determine address of
default interface.  This may happen because there is no default route
configured.  Without a default route, an interface must be: Network is
unreachable
Aug 31 20:41:49 localhost omdiscd[1290]: Unable to initialize network.
Exiting.
```

An example of a correct routing is below

```
[root@localhost log]# route −n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         10.0.0.99       0.0.0.0         UG    0      0        0 eth0
```

Basically from now on everything will get easier. Just start

```
omdiscd
```

And have a look at your logfiles you should see something similar to this

```
Sep  2 10:00:49 oscar0 kernel: openMosix configuration changed: This is openMosix #2780 (of 6 co
Sep  2 10:00:49 oscar0 kernel: openMosix #2780 is at IP address 192.168.10.220
Sep  2 10:00:49 oscar0 kernel: openMosix #2638 is at IP address 192.168.10.78
Sep  2 10:00:49 oscar0 kernel: openMosix #2646 is at IP address 192.168.10.86
Sep  2 10:00:49 oscar0 kernel: openMosix #2627 is at IP address 192.168.10.67
Sep  2 10:00:49 oscar0 kernel: openMosix #2634 is at IP address 192.168.10.74
```

Congratulations , your openMosix cluster is now working.

omdiscd has some other options that you can use. You can either run omdiscd as a daemon (default) or in the foreground where output goes to the screen (standard output) omdiscd −n . An interface can be specified with the −i option.

Now lets still have a short look at the other tool , it's showmap. This tool will show you the newly auto generated openMosix map.

```
[root@oscar0 root]# showmap
My Node−Id: 0x0adc

Base Node−Id Address          Count
−−−−−−−−−−−− −−−−−−−−−−−−−−−− −−−−−
0x0adc       192.168.10.220   1
0x0a4e       192.168.10.78    1
```

```
0x0a56       192.168.10.86    1
0x0a43       192.168.10.67    1
0x0a4a       192.168.10.74    1
```

Auto−discovery has some other features not listed here such as a routing mechanism for clusters with more than one network. More detailed information can be found in the README and DESIGN files in the user−land tools source tree.

More recent versions of the openMosix rc scripts will first verify wether an /etc/openmosix.map file or similar exists before trying to use autoconfiguration.

5.2. Compiling auto−discovering

If you are compiling autodiscovery from source you will need to make a small modification to openmosix.c. One of the first lines will be

```
#define ALPHA
```

You will need to put this in comment. If you want to have some more logging available to you you should edit main.c to show log_set_debug(DEBUG_TRACE_ALL); (somewhere around line 84) now run

```
% make clean
% make
```

5.3. Troubleshooting autodiscovery

Sometimes however autodiscovery does not function as you would like, for example a node might not see multicast traffic from other nodes. This has occurred with some PCMCIA ethernet drivers. One solution is to place the interface in promiscuous and or multicast mode as detailed below:

```
Aug 31 20:45:58 localhost kernel: openMosix configuration changed: This is openMosix #98 (of 1 co
Aug 31 20:45:58 localhost kernel: openMosix #98 is at IP address 10.0.0.98Aug 31 20:45:58 localho
openMosix  Aug 31 20:45:58 localhost kernel: Received an unauthorized information request from 10
```

What you should to then is try to force your NIC into promiscuous and/ or multicast mode manually.

```
ifconfig ethx promisc
or
ifconfig ethx multicast
```

You might also want to run

```
tcpdump −i eth0 ether multicast
```

which will have the same effect but you will now also be able to see the packages yourself.

On some Layer 3 switches you other configs might be required. An openMosix user found out that on his Switch Summit48Si (Extreme Networks) he had to run

```
disable ipmcforwarding (to deactivate the routing of multicast paquets)
disable igmp snooping
```

before he was the different omdiscd's were able to see eachother, other switches might require similar configs.

```
Aug 31 22:14:43 inspon omdiscd[1422]: Simulated notification to activate openMosix
[root@inspon root]# showmap
My Node−Id: 0x0063

Base Node−Id Address          Count
−−−−−−−−−−−− −−−−−−−−−−−−−−−− −−−−−
0x0063       10.0.0.99        1
[root@inspon root]# /etc/init.d/openmosix status
OpenMosix is currently disabled
[root@inspon root]#
```

If you see the *simulated* you have probably forgotten to put the

```
#define ALPHA
```

in comment.

I have also noticed that autodiscovery does not work with FireWire based network cards.

Chapter 6. PlumpOS

6.1. What Plump/OS is

PlumpOS is a CD−based GNU/Linux/openMosix mini−distribution designed to allow users to quickly, or temporarily, add nodes to an openMosix cluster; as I write this in march 2003 the version (release 6.9 RC1) is a 16.7M ISO download.

This chapter is a quick hack up by Peter Willis of a very similar chapter contributed by Jean−David Marrow (who is the author of Clump/OS and inspirer of PlumpOS − props to Jean−David for his fine work on the departed Clump/OS).

6.2. How does it work?

First of all, set up your machine's BIOS in order to make it boot from CD. Check your motherboard's manual to see how to set up this. Upon booting up you will receive a prompt to boot PlumpOS using a specific kernel; several ones have been provided for you but, you also have the option of providing your own kernel + modules. This will be explained later in the ``Getting Started'' section. The boot menu should tell you all the kernels available for you to use; simply type the name of one (and optionally some kernel arguments) and press return/enter.

At boot−time, PlumpOS will auto−probe for network cards and, if any gets detected, it'll try and configure them via DHCP. If successful, it will fire up omdiscd on each interface a DHCP lease is received on. Currently this has only been tested in PCs with 1 network card so YMMV.

It works for myself, but may not work for you; if you experience difficulties, please email me with as much information about your system as possible −− after you have investigated the problem. (Check the sourceforge homepage for PlumpOS's mailing list, sign up and post your question with as much detailed information as possible. The openMosix−general and openMosix−devel lists are NOT for PlumpOS problems, they are for openMosix−specific problems, and if you can't tell the difference just send it to the PlumpOS list or my contact email).

6.3. Requirements

As the purpose of PlumpOS is to add nodes to a cluster, it is assumed that you already have a running openMosix cluster −− or perhaps only a single openMosix node −− from which you will be initiating jobs. All machines in the cluster must conform to the following requirements:

- PlumpOS Machine(s) 586+ CPU
- bootable CD−ROM drive
- Network Interface Card
- Between 32M and 128M of RAM. Because of some oddities in the openMosix kernels at the time of this writing, simply passing the size of the ramdisk is not working so one has to actually use a ramdisk pre−made for a specific size. There should be several ramdisks avaliable at PlumpOS mirrors which can be placed in the disks' root directory on the actual cdrom ISO (or rootdisk/disks/ if you're inside the PlumpOS package directory).
- Master Machine(s) GNU/Linux/openMosix kernel (same version as all the PlumpOS kernel you are booting on the Child/Slave PlumpOS Machine(s))

- Network Environment Running DHCP server (if you don't, or won't, run DHCP, you can still manually configure your system: simply go to each PlumpOS machine and enter in your desired configuration information with the supplied networking tools. Using DHCP is highly recommended however, and will greatly simplify your life in the long run.

The following network modules are present in most if not all of the supplied kernels' modules tarball (in /kernels/KERNELNAME/modules.tgz), although not all support auto−probing; if you don't see support for your card in this list, then PlumpOS will not work for you.

3c501.o, 3c503.o, 3c505.o, 3c507.o, 3c509.o, 3c515.o, 3c59x.o, 8139cp.o, 8139too.o, 82596.o, ac3200.o, acenic.o, aironet4500_card.o, aironet4500_core.o, aironet4500_proc.o, arlan−proc.o, arlan.o, at1700.o, bsd_comp.o, cs89x0.o, de4x5.o, depca.o, dgrs.o, dl2k.o, dmfe.o, dummy.o, e100/e100.o, e1000/e1000.o, e2100.o, eepro.o, eepro100.o, eexpress.o, epic100.o, eth16i.o, ewrk3.o, fealnx.o, hamachi.o, hp−plus.o, hp.o, hp100.o, lance.o, lp486e.o, mii.o, natsemi.o, ne.o, ne2k−pci.o, ni5010.o, ni52.o, ni65.o, ns83820.o, pcmcia, pcnet32.o, ppp_async.o, ppp_deflate.o, ppp_generic.o, ppp_synctty.o, pppoe.o, pppox.o, sis900.o, sk98lin/sk98lin.o, slhc.o, smc−ultra.o, smc9194.o, starfire.o, strip.o, sundance.o, sungem.o, sunhme.o, tc35815.o, tg3.o, tlan.o, tokenring/{3c359.o abyss.o ibmtr.o lanstreamer.o olympic.o smctr.o tms380tr.o tmsisa.o tmspci.o}, tulip/tulip.o, via−rhine.o, wavelan.o, wd.o, winbond−840.o, wireless/{airo.o airo_cs.o hermes.o orinoco.o orinoco_cs.o orinoco_pci.o orinoco_plx.o}, yellowfin.o

Please also note that PlumpOS may not work on a laptop: it definitely doesn't support PCMCIA cards (yet), and will probably not configure openMosix properly if your machine contains multiple connected Ethernet adapters. This is a temporary limitation of the configuration scripts, and should be resolved in future releases.

6.4. Getting Started

You can download the latest PlumpOS package under the terms of the GPL, without warranty of any kind, from any PlumpOS mirror. Upon downloading, unpack the archive

```
$ tar −xvzf plumpos−6.9−rc1.tar.gz
```

and enter the new directory "plumpos−6.9−rc1/". Now you have several options as to how you go about setting up PlumpOS.

First let's familiarize with the directory structure here. There should be 3 directories:

- rootdisk, which contains the layout of the iso to be created;
- scripts, which contains small programs used to help in the creation of the iso;
- final, which will contain the ISO generated by the whole process.

There should also be a file called install which you will run when you are done configuring your system.

If you intend on using your own kernel with PlumpOS, there are several steps you must follow in order for it to boot properly. First you must make a new directory in the rootdisk/kernels/ directory that will be in the old and often blamed DoS 8.3 character format and only contain letters and numbers (and a single '.'). In that directory you should put a file named bzImage which is your kernel and a file named modules.tgz which is a gzipped tarball of your kernel's modules (with the relative path of lib/modules/ so it can be extracted from the root dir). Optionally you can also provide the System.map and config file for your kernel. When this is done you may simply run the install program and it will detect and install your kernel(s) for use with the generated ISO.

Now, say you want to include a 3rd party add−on package. This is done very simply: create a gzipped tarball containing all the files you want to include in your package (relative to "/") and put these packages into the rootdisk/packages/ directory. Then edit the file rootdisk/packages/list and add the path relative to / on the ramdisk where the package can be located (in other words for a package named "openssl.tar.gz", add the line "/cdrom/packages/openssl.tar.gz" to the file rootdisk/packages/list). Optionally you can use a line such as "cdrom:openssl.tar.gz" which will automatically search the cdrom's packages directory for the package "openssl.tar.gz". In future releases this will be useful for things like nfs and boot floppies, so for now don't worry about it ;)

Chapter 7. Cluster Installation

7.1. Cluster Installations

This chapter does not deal with installing openMosix as such, it does however deal with installing multiple machines with openMosix. Automated or semi automated mass installs.

7.2. DSH, Distributed Shell

At the time of this writing (May 2003) DSH's most current release is available from *http://www.netfort.gr.jp/~dancer/software/downloads/* More info on the package can be found on *http://www.netfort.gr.jp/~dancer/software/dsh.html* The latest version available for download is 0.23.6 You will need both libdshconfig−0.20.8.tar.gz and dsh−0.23.5.tar.gz Start with installing libdshconfig

```
./configure
make
make install
```

Repeat the process for the dsh package.

Say we have a small cluster with a couple of nodes. To make life easier we want type each command once but have it executed on each node. You then have to create a file in $HOME/.dsh/group/clusterwname that lists the ip's of your cluster. eg.

```
[root@inspon root]# cat .dsh/group/mosix
192.168.10.220
192.168.10.84
```

As an example we run ls on each of these machines We use −g to use the mosix group (this way you can create subsets of a group with different configurations)

```
[root@inspon root]# dsh −r ssh −g mosix ls
192.168.10.84: anaconda−ks.cfg
192.168.10.84: id_rsa.pub
192.168.10.84: install.log
192.168.10.84: install.log.syslog
192.168.10.84: openmosix−kernel−2.4.17−openmosix1.i686.rpm
192.168.10.84: openmosix−tools−0.2.0−1.i386.rpm
192.168.10.220: anaconda−ks.cfg
192.168.10.220: id_dsa.pub
192.168.10.220: id_rsa.pub
192.168.10.220: openmosix−kernel−2.4.17−openmosix1.i686.rpm
192.168.10.220: openmosix−tools−0.2.0−1.i386.rpm
192.168.10.220: oscar−1.2.1rh72
192.168.10.220: oscar−1.2.1rh72.tar.gz
```

Note that neither of the machines ask for a password. This is because we have set up RSA authentication between the different accounts. If you want to run commands with multiple parameters you will either have to put the command between quotes.

```
[root@inspon root]# dsh −r ssh −g mosix "uname −a"
192.168.10.84: Linux omosix2.office.be.stone−it.com 2.4.17−openmosix1 #1
Wed May 29 14:32:28 CEST 2002 i686 unknown
192.168.10.220: Linux oscar0 2.4.17−openmosix1 #1 Wed May 29 14:32:28 CEST
```

```
2002 i686 unknown
```

or use the −c −− option. Both give basically the same output.

```
[root@inspon root]# dsh −r ssh −g mosix −c −− uname −a
192.168.10.220: Linux oscar0 2.4.17−openmosix1 #1 Wed May 29 14:32:28 CEST
2002 i686 unknown
192.168.10.84: Linux omosix2.office.be.stone−it.com 2.4.17−openmosix1 #1
Wed May 29 14:32:28 CEST 2002 i686 unknown
```

III. Administrating openMosix

Table of Contents

Chapter 8. Administrating openMosix

8.1. Basic Administration

openMosix provides the advantage of process migration to HPC−applications. The administrator can configure and tune the openMosix−cluster by using the openMosix−user−space−tools or the /proc/hpc interface which will be now described in detail.

Up till openMosix version 2.4.16 the /proc interface was named /proc/mosix ! Until openMosix version 2.4.17 it was named /proc/hpc.

8.2. Configuration

The values in the flat files in the /proc/hpc/admin directory presenting the current configuration of the cluster. Also the administrator can write its own values into these files to change the configuration during runtime, e.g.

Table 8−1. Changing /proc/hpc parameters

echo 1 > /proc/hpc/admin/block	blocks the arrival of remote processes
echo 1 > /proc/hpc/admin/bring	bring all migrated processes home

...

Table 8−2. /proc/hpc/admin/

(binary files)	config	the main configuration file (written by the setpe util)
(flat files)	block	allow/forbid arrival of remote processes
	bring	bring home all migrated processes
	dfsalinks	list of current symbolic dfsa−links
	expel	sending guest processes home
	gateways	maximum number of gateways
	lstay	local processes should stay
	mospe	contains the openMosix node id
	nomfs	disables/enables MFS
	overheads	for tuning
	quiet	stop collecting load−load−balancing informations
	decay−interval	interval for collecting informations about load−balancing
	slow−decay	default 975
	fast−decay	default 926
	speed	speed relative to PIII/1GHz)
	stay	enables/disables automatic process migration

Table 8−3. Writing a 1 to the following files /proc/hpc/decay/

clear	clears the decay statistics
cpujob	tells openMosix that the process is cpu−bound
iojob	tells openMosix that the process is io−bound
slow	tells openMosix to decay its statistics slow
fast	tells openMosix to decay its statistics fast

Table 8−4. Informations about the other nodes

/proc/hpc/nodes/[openMosix_ID]/CPUs	how many CPU's the node has
/proc/hpc/nodes/[openMosix_ID]/load	the openMosix load of this node
/proc/hpc/nodes/[openMosix_ID]/mem	available memory as openMosix believes
/proc/hpc/nodes/[openMosix_ID]/rmem	available memory as Linux believes
/proc/hpc/nodes/[openMosix_ID]/speed	speed of the node relative to PIII/1GHz
/proc/hpc/nodes/[openMosix_ID]/status	status of the node
/proc/hpc/nodes/[openMosix_ID]/tmem	available memory
/proc/hpc/nodes/[openMosix_ID]/util	utilization of the node

Table 8−5. Additional Informations about local processes

/proc/[PID]/cantmove	reason why a process cannot be migrated
/proc/[PID]/goto	to which node the process should migrate
/proc/[PID]/lock	if a process is locked to its home node
/proc/[PID]/nmigs	how many times the process migrated
/proc/[PID]/where	where the process is currently being computed
/proc/[PID]/migrate	same as goto remote processes
/proc/hpc/remote/from	the home node of the process
/proc/hpc/remote/identity	additional informations about the process
/proc/hpc/remote/statm	memory statistic of the process
/proc/hpc/remote/stats	cpu statistics of the process

8.3. the userspace−tools

These following tools are providing easy administration to openMosix clusters.

```
migrate −send a migrate request to a process
                syntax:
                        migrate [PID] [openMosix_ID]
```

```
mon             −is a ncurses−based terminal monitor
                 several informations about the current status are displayed in bar−charts
```

```
mosctl          −is the openMosix main configuration utility
                syntax:
                        mosctl  [stay|nostay]
                                [lstay|nolstay]
                                [block|noblock]
                                [quiet|noquiet]
                                [nomfs|mfs]
                                [expel|bring]
                                [gettune|getyard|getdecay]

                        mosctl  whois   [openMosix_ID|IP−address|hostname]

                        mosctl  [getload|getspeed|status|isup|getmem|getfree|getutil]   [openMosi

                        mosctl  setyard [Processor−Type|openMosix_ID||this]

                        mosctl  setspeed        interger−value

                        mosctl  setdecay interval       [slow fast]
```

Table 8−6. more detailed

stay	no automatic process migration
nostay	automatic process migration (default)
lstay	local processes should stay
nolstay	local processes could migrate
block	block arriving of guest processes
noblock	allow arriving of guest processes
quiet	disable gathering of load−balancing informations
noquiet	enable gathering of load−balancing informations
nomfs	disables MFS
mfs	enables MFS
expel	send away guest processes
bring	bring all migrated processes home
gettune	shows the current overhead parameter
getyard	shows the current used Yardstick
getdecay	shows the current decay parameter
whois	resolves openMosix−ID, ip−addresses and hostnames of the cluster
getload	display the (openMosix−) load
getspeed	shows the (openMosix−) speed
status	displays the current status and configuration
isup	is a node up or down (openMosix kind of ping)
getmem	shows logical free memory
getfree	shows physical free mem
getutil	display utilization
setyard	sets a new Yardstick−value
setspeed	sets a new (openMosix−) speed value

setdecay	sets a new decay−interval

```
mosrun          -run a special configured command on a chosen node
                syntax:
                        mosrun  [−h|openMosix_ID| list_of_openMosix_IDs] command [arguments]
```

The mosrun command can be executed with several more commandline options. To ease this up there are several preconfigured run−scripts for executing jobs with a special (openMosix) configuration.

Table 8−7. extra options for mosrun

nomig	runs a command which process(es) won't migrate
runhome	executes a command locked to its home node
runon	runs a command which will be directly migrated and locked to a node
cpujob	tells the openMosix cluster that this is a cpu−bound process
iojob	tells the openMosix cluster that this is a io−bound process
nodecay	executes a command and tells the cluster not to refresh the load−balancing statistics
slowdecay	executes a command with a slow decay interval for collecting load−balancing statistics
fastdecay	executes a command with a fast decay interval for collecting load−balancing statistics

```
setpe           −manual node configuration utility
                syntax:
                        setpe   −w −f   [hpc_map]
                        setpe   −r [−f  [hpc_map]]
                        setpe   −off

−w reads the openMosix configuration from a file (typically /etc/hpc.map)
−r writes the current openMosix configuration to a file (typically /etc/hpc.map)
−off turns the current openMosix configuration off
```

```
tune            openMosix calibration and optimizations utility.
                (for further informations review the tune−man page)
```

Additional to the /proc interface and the commandline−openMosix utilities (which are using the /proc interface) there is a patched "ps" and "top" available (they are called "mps" and "mtop") which displays also the openMosix−node ID on a column. This is useful for finding out where a specific process is currently being computed.

This actually summarised the command line tools, but have a look at openMosixview which is a GUI for the most common administration tasks, and which ill be discussed in a future chapter.

8.4. Cluster Mask

(by Moshe Bar)

Several people have asked for a feature in openMosix which allows to specifiy to which nodes a given process and it's children can migrate and to which nodes it cannot.

Simone Ettore has just committed a new patch to the CVS which allows you to do just that.

Here is how it works:

- /proc/[pid]/migfilter enable/disable the capability of filter migration.
- /proc/[pid]/mignodes is a bit−list of nodes. The bit position of a node is calculated as 2^(PE−1). PE is node number.
- /proc/[pid]/migpolicy is the policy of the filtering: 0=DENY: the process can migrate in all nodes except when the relative bit on mignodes is 1 1=ALLOW: the process can migrate in all nodes where the relative bit on mignodes is 1

We are shortly going to release also a simple user−land tool to set the node mask, but I would like you guys to give it a try asap before we release it as openMosix 2.4.20−3.

Chapter 9. Tuning Mosix

9.1. Introduction

Some of the parts below are still from the old Mosix Howto, as time passes these parts will get replaced by relevant openMosix parts, however some things are still the same , but your mileage may vary.

9.2. Creating a "Master" node

Although openMosix architcture does not require a master node as such, you might want to have a head node from where you launch processes, this might be a multihomed node from where users log in to your cluster. You want to configure your machine to make processes migrate away

You have to trick the node in thinking it is the slowest node around and it'd better migrate all it's processes to the faster nodes.

You will have to make it "slow" with :

```
mosctl setspeed [n]
```

where n should be much lower than the speed of the other nodes Processes will move/migrate away fast. You can get the speed of a node with :

```
mosctl getspeed
```

9.3. Optimizing Mosix

Editorial Comment: To be checked with openMosix versions

Login a normal terminal as root. Type

```
       setpe −r
```

which, if everything went right, will give you a listing of your /etc/mosix.map. If things did not go right, try

```
        setpe −w −f /etc/mosix.map
```

to set up your node. Then, type

```
       cat /proc/$$/lock
```

to see if your child processes are locked in your mode (1) or can migrate (0). If for some reason you find your processes are locked, you can change this with

```
        echo 0 > /proc/$$/lock
```

until you fix the problem. Repeat the whole configuration scheme for a second computer. The programs tune_kernel and prep_tune that Mosix uses to calibrate the individual nodes do not work with the SuSE distribution. However, you can fake it. First, bring the computer you want to tune and another computer with Mosix installed down to single user mode by typing

```
        init 1
```

as root. All other computers on the network should be shutdown if possible. On both machines, run the following commands:

```
        /etc/init.d/network start
        /etc/init.d/mosix start
        echo 1 > /proc/mosix/admin/quiet
```

This fakes prep_tune and the first parts of tune_kernel. Note that if you have a laptop with a pcmcia network card, you will have to run

```
        /etc/init.d/pcmcia start
```

instead of "/etc/init.d/network start". On the computer you want to tune, run tune_kernel and follow instructions. Depending on your machines, this can take a while − if you have a dog, this might be the time to go on that long, long walk you've always promised him. tune_kernel will create a program called "pg" in /root for testing reasons. Ignore it. After tuning is over, copy the contents of /tmp/overheads to the file /etc/overheads (and/or recompile the kernel). Repeat the tuning procedure for each computer. Reboot, enjoy Mosix, and don't forget to brag to your friends about your new cluster.

9.4. Channel Bonding Made Easy

Contributed by Evan Hisey

Channel bonding is actually horrible easy. This may explain the lack of documentation on this subject A bonded network appears as a normal network to the applications. All machines on a subnet must be bonded the same way. Bonded and non−bonded machine really don't talk well to each other.

Channel bonding needs at least two physical sub−nets but can have more(Currently I have a tri−bonded cluster). To enable bonding you need to either compile in to the kernel or as a module (bonding.o) the Channel Bonding kernel code, as of 2.4.x is it a standard option of the kernel. The NIC's are setup as normal with except that you only us 'ifconfig' to initialize the first card of the bond. 'ifenslave' is used to initialize the remaining cards in the bonded connection. 'ifenslave' can be locate in the linux/Documentation/network/ directory. It will need to be compiled as it is a .c file. The basic format for use is

```
ifenslave <master> <slave1> <slave2> ...
```

Channel bonded networks can connect to standard networks via a router or bridge that supports channel bonding(I just use an extra NIC and port−forwarding in the head node).

9.5. Updatedb

Updatedb in combination with mfs can cause some issues, you might want to add /mfs to the PRUNEFPATHS or mfs to the PRUNEFS in your /etc/updatedb.conf to disable updatedb from indexing this mountpoints.

9.6. openMosix and FireWire

openMosix does gain performance by using another type of network device, as described within the paper about *openMosix and FireWire*

Chapter 10. openMosixview

10.1. Introduction

openMosixview is the next version and a complete rewrite of Mosixview. It is a cluster−management GUI for openMosix−cluster and everybody is invited to download and use it (at your own risk and responsibility). The openMosixview−suite contains 5 useful applications for monitoring and administrating openMosix−cluster.

openMosixview the main monitoring+administration application
openMosixprocs a process−box for managing processes
openMosixcollector collecting daemon which logs cluster+node informations
openMosixanalyzer for analyzing the data collected by the openMosixcollector
openMosixhistory a process−history for your cluster

All parts are accessible from the main application window. The most common openMosix−commands are executable by a few mouse−clicks. An advanced execution dialog helps to start applications on the cluster. "Priority−sliders" for each node simplifying the manual and automatic load−balancing. openMosixview is now adapted to the openMosix−auto−discovery and gets all configuration−values from the openMosix /proc−interface.

10.2. openMosixview vs Mosixview

openMosixview is fully designed for openMosix cluster only. The Mosixview−website (and all mirrors) will stay as they are but all further developing will continue with openMosixview located at the new domain *www.openmosixview.com*

If you have: questions, features wanted, problems during installation, comments, exchange of experiences etc. feel free to mail me, Matt Rechenburg or subscribe to the openMosix/Mosixview−mailing−list and mail to the openMosix/Mosixview−mailing−list

changes: (to Mosixview 1.1) openMosixview is a complete rewrite "from the scratch" of Mosixview! It has the same functionalities but there are fundamental changes in ALL parts of the openMosixview source−code. It is tested with a constantly changing cluster topography (required for the openMosix auto−discovery) All "buggy" parts are removed or rewritten and it (should ;) run much more stable now.

adapted to the openMosix−auto−discovery
not using /etc/mosix.map or any cluster−map file anymore
removed the (buggy) map−file parser
rewrote all parts/functions/methods to a cleaner c++ interface
fixed some smaller bugs in the display
replaced MosixMem+Load with the openMosixanalyzer
... many more changes

10.3. Installation

Requirements

- QT library
- root rights !
- rlogin and rsh (or ssh) to all cluster−nodes without password the openMosix userland−tools mosctl, migrate, runon, iojob, cpujob ... (download them from the www.openmosix.org website)

On a RH 8.0 you will need at least the following rpm's qt−3.0.5−17, libmng−1.0.4, XFree86−Mesa−libGLU−4.2.0, glut−3.7 etc ...

Documentation about openMosixview There is a full HTML−documentation about openMosixview included in every package. You find the startpage of the documentation in your openMosixview installation directory: openmosixview/openmosixview/docs/en/index.html

The RPM−packages have their installation directories in: /usr/local/openmosixview

---

10.3.1. Installation of the RPM−distribution

Download the latest version of openMosixview rpm−package. Then just execute e.g.:

```
rpm −i openmosixview−1.4.rpm
```

This will install all binaries in /usr/bin To uninstall:

```
rpm −e openmosixview
```

10.3.2. Installation of the source−distribution

Download the latest version of openMosixview and unzip+untar the sources and copy the tarball to e.g. /usr/local/.

```
gunzip openmosixview−1.4.tar.gz
tar −xvf openmosixview−1.4.tar
```

10.3.3. Automatic setup−script

Just cd to the openmosixview−directory and execute

```
./setup [your_qt_2.3.x_installation_directory]
```

10.3.4. Manual compiling

Set the QTDIR−Variable to your actual QT−Distribution, e.g.

```
export QTDIR=/usr/lib/qt−2.3.0 (for bash)
or
```

```
setenv QTDIR /usr/lib/qt-2.3.0 (for csh)
```

10.3.5. Hints

(from the testers of openMosixview/Mosixview who compiled it on different linux−distributions, thanks again) Create the link /usr/lib/qt pointing to your QT−2.3.x installation e.g. if QT−2.3.x is installed in /usr/local/qt−2.3.0

```
ln -s /usr/local/qt-2.3.0 /usr/lib/qt
```

Then you have to set the QTDIR environment variable to

```
export QTDIR=/usr/lib/qt (for bash)
or
setenv QTDIR /usr/lib/qt (for csh)
```

After that the rest should work fine:

```
./configure
make
```

then do the same in the subdirectory openmosixcollector, openmosixanalyzer, openmosixhistory and openmosixviewprocs. Copy all binaries to /usr/bin

```
cp openmosixview/openmosixview /usr/bin
cp openmosixviewproc/openmosixviewprocs/mosixviewprocs /usr/bin
cp openmosixcollector/openmosixcollector/openmosixcollector /usr/bin
cp openmosixanalyzer/openmosixanalyzer/openmosixanalyzer /usr/bin
cp openmosixhistory/openmosixhistory/openmosixhistory /usr/bin
```

And the openmosixcollector init−script to your init−directory e.g.

```
cp openmosixcollector/openmosixcollector.init /etc/init.d/openmosixcollector
or
cp openmosixcollector/openmosixcollector.init /etc/rc.d/init.d/openmosixcollector
```

Now copy the openmosixprocs binary on each of your cluster−nodes to /usr/bin/openmosixprocs

```
rcp openmosixprocs/openmosixprocs your_node:/usr/bin/openmosixprocs
```

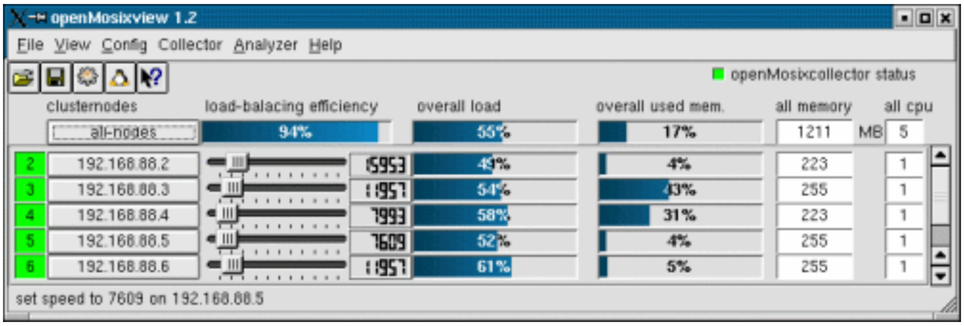
You can now execute mosixview

```
openmosixview
```

10.4. using openMosixview

10.4.1. main application

Here is a picture of the main application−window. The functionality is explained in the following.


openMosixview 1.2
File View Config Collector Analyzer Help
openMosixcollector status

	clusternodes	load-balacing efficiency		overall load	overall used mem.	all memory		all cpu
	all-nodes	94%		55%	17%	1211	MB	5
2	192.168.88.2		15953	49%	4%	223		1
3	192.168.88.3		11957	54%	43%	255		1
4	192.168.88.4		7993	58%	31%	223		1
5	192.168.88.5		7609	52%	4%	255		1
6	192.168.88.6		11957	61%	5%	255		1

set speed to 7609 on 192.168.88.5

openMosixview displays a row with a lamp, a button, a slider, a lcd−number, two progress−bars and some labels for each cluster−member. The lights at the left are displaying the openMosix−Id and the status of the cluster−node. Red if down, green for available.

If you click on a button displaying the ip−address of one node a configuration−dialog will pop up. It shows buttons to execute the most common used "mosctl"−commands. (described later in this HOWTO) With the "speed−sliders" you can set the openMosix−speed for each host. The current speed is displayed by the lcd−number.

You can influence the load−balancing of the whole cluster by changing these values. Processes in a openMosix−Cluster are migrating easier to a node with more openMosix−speed than to nodes with less speed. Sure it is not the physically speed you can set but it is the speed openMosix "thinks" a node has. e.g. a cpu−intensive job on a cluster−node which speed is set to the lowest value of the whole cluster will search for a better processor for running on and migrate away easily.

The progress bars in the middle gives an overview of the load on each cluster−member. It displays in percent so it does not represent exactly the load written to the file /proc/hpc/nodes/x/load (by openMosix), but it should give an overview.

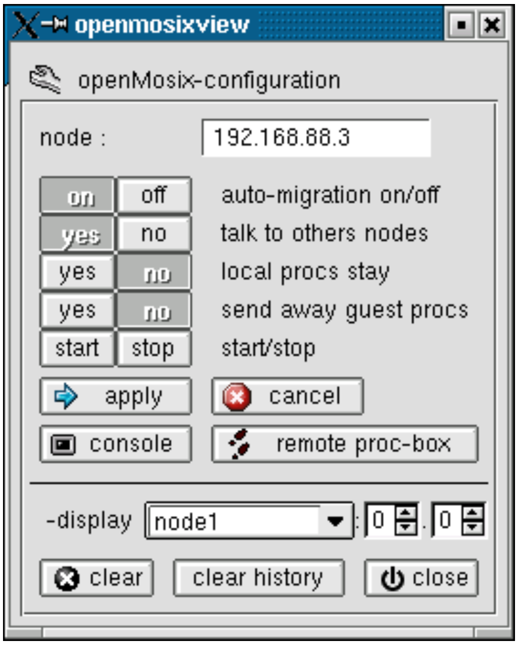
The next progressbar is for the used memory the nodes. It shows the currently used memory in percent from the available memory on the hosts (the label to the right displays the available mem). How many CPUs your cluster have is written in the box to the right. The first line of the main windows contains a configuration button for "all−nodes". You can configure all nodes in your cluster similar by this option.

How good the load−balancing works is displayed by the progressbar in the top left. 100% is very good and means that all nodes nearly have the same load.

Use the collector− and analyzer−menu to manage the openMosixcollector and open the openMosixanalyzer. This two parts of the openMosixview−application suite are useful for getting an overview of your cluster during a longer period.

10.4.2. the configuration−window

This dialog will pop up if an "cluster−node"−button is clicked.



openMosix-configuration

node : 192.168.88.3

[✓ on] off auto-migration on/off
[✓ yes] no talk to others nodes
yes [✓ no] local procs stay
yes [✓ no] send away guest procs
start stop start/stop

-display node1 : 0 . 0

The openMosix−configuration of each host can be changed easily now. All commands will be executed per "rsh" or "ssh" on the remote hosts (even on the local node) so "root" has to "rsh" (or "ssh") to each host in the cluster without prompting for a password (it is well described in a Beowulf documentation or on the HOWTO on this page how to configure it).

The commands are:

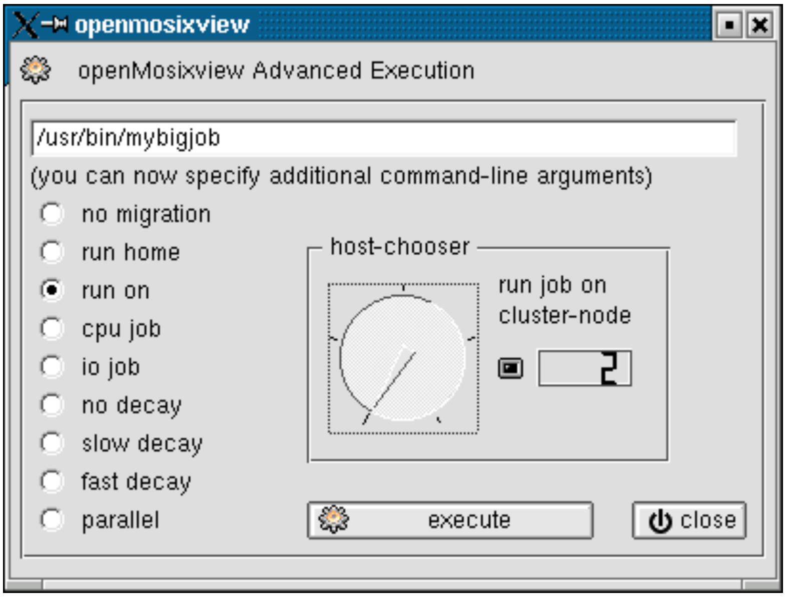
```
automigration on/off
quiet yes/no
bring/lstay yes/no
exspel yes/no
openMosix start/stop
```

If openMosixprocs is properly installed on the remote cluster−nodes click the "remote proc−box"−button to open openMosixprocs (proc−box) from remote. xhost +hostname will be set and the display will point to your localhost. The client is executed on the remote also per "rsh" or "ssh". (the binary openmosixprocs must be copied to e.g. /usr/bin on each host of the cluster) openMosixprocs is a process−box for managing your programs. It is useful to manage programs started and running local on the remote nodes and is described later in this HOWTO.

If you are logged on your cluster from a remote workstation insert your local hostname in the edit−box below the "remote proc−box". Then openMosixprocs will be displayed on your workstation and not on the cluster−member you are logged on. (maybe you have to set "xhost +clusternode" on your workstation). There is a history in the combo−box so you have to write the hostname only once.

10.4.3. advanced−execution

If you want to start jobs on your cluster the "advanced execution"−dialog may help you.



/usr/bin/mybigjob

(you can now specify additional command-line arguments)

- ○ no migration
- ○ run home
- ● run on
- ○ cpu job
- ○ io job
- ○ no decay
- ○ slow decay
- ○ fast decay
- ○ parallel

run job on cluster-node 2

Choose a program to start with the "run−prog" button (file−open−icon) and you can specify how and where the job is started by this execution−dialog. There are several options to explain.

10.4.4. the command−line

You can specify additional commandline−arguments in the lineedit−widget on top of the window.

Table 10−1. how to start

−no migration	start a local job which won't migrate
−run home	start a local job
−run on	start a job on the node you can choose with the "host−chooser"
−cpu job	start a computation intensive job on a node (host−chooser)
−io job	start a io intensive job on a node (host−chooser)
−no decay	start a job with no decay (host−chooser)
−slow decay	start a job with slow decay (host−chooser)
−fast decay	start a job with fast decay (host−chooser)
−parallel	start a job parallel on some or all node (special host−chooser)

10.4.5. the host−chooser

For all jobs you start non−local simple choose a host with the dial−widget. The openMosix−id of the node is also displayed by a lcd−number. Then click execute to start the job.

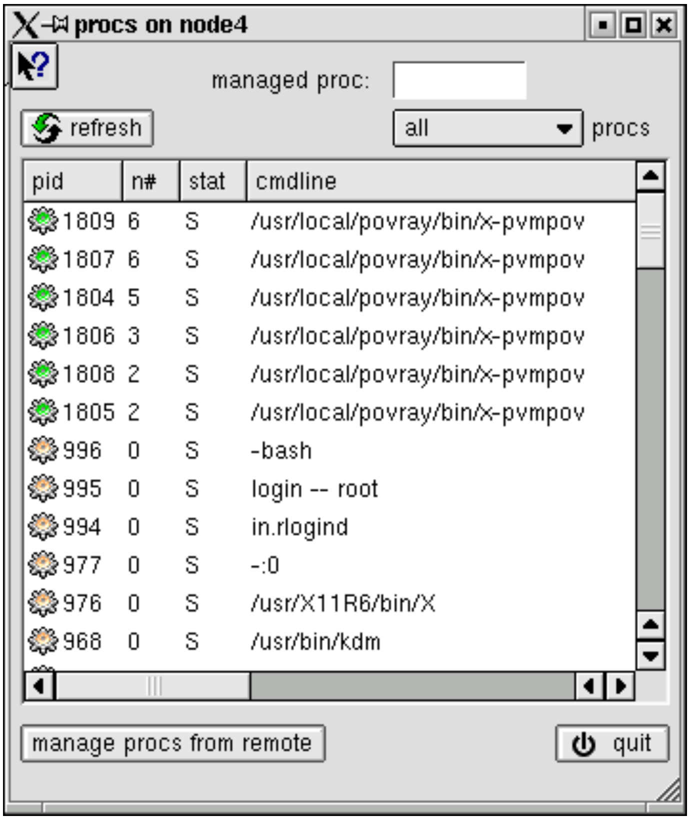
10.4.6. the parallel host−chooser

You can set the first and last node with 2 spinboxes. Then the command will be executed an all nodes from the first node to the last node. You can also inverse this option.

10.5. openMosixprocs

10.5.1. intro

This process−box is really useful for managing the processes running on your cluster.



managed proc:

all procs

pid	n#	stat	cmdline
1809	6	S	/usr/local/povray/bin/x-pvmpov
1807	6	S	/usr/local/povray/bin/x-pvmpov
1804	5	S	/usr/local/povray/bin/x-pvmpov
1806	3	S	/usr/local/povray/bin/x-pvmpov
1808	2	S	/usr/local/povray/bin/x-pvmpov
1805	2	S	/usr/local/povray/bin/x-pvmpov
996	0	S	-bash
995	0	S	login -- root
994	0	S	in.rlogind
977	0	S	-:0
976	0	S	/usr/X11R6/bin/X
968	0	S	/usr/bin/kdm

You should install it on every cluster−node!

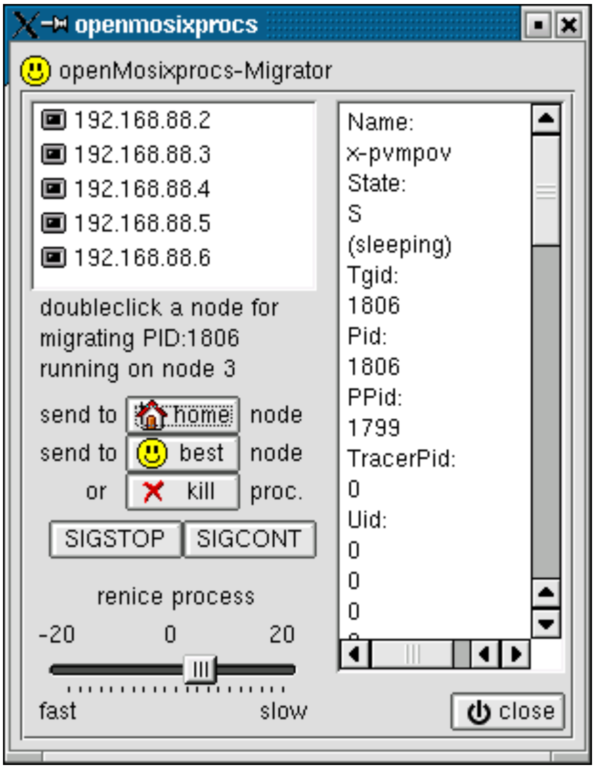
The processlist gives an overview what is running where. The second column displays the openMosix−node ID of each process. 0 means local, all other values are remote nodes. Migrated processes are marked with a green icon and non movable processes have a lock.

By double−clicking a process from the list the migrator−window will pop−up for managing e.g. migrating the process. There are also options to migrate the remote processes away, send SIGSTOP and SIGCONT to it or to "renice" it.

If you click on the "manage procs from remote" button a new window will come up (the remote−procs windows) displaying the process currently migrated to this host.

10.5.2. the migrator−window

This dialog will pop up if process from the process box is clicked.



192.168.88.2
192.168.88.3
192.168.88.4
192.168.88.5
192.168.88.6

doubleclick a node for migrating PID:1806 running on node 3

Name:
x-pvmpov
State:
S
(sleeping)
Tgid:
1806
Pid:
1806
PPid:
1799
TracerPid:
0
Uid:
0
0
0

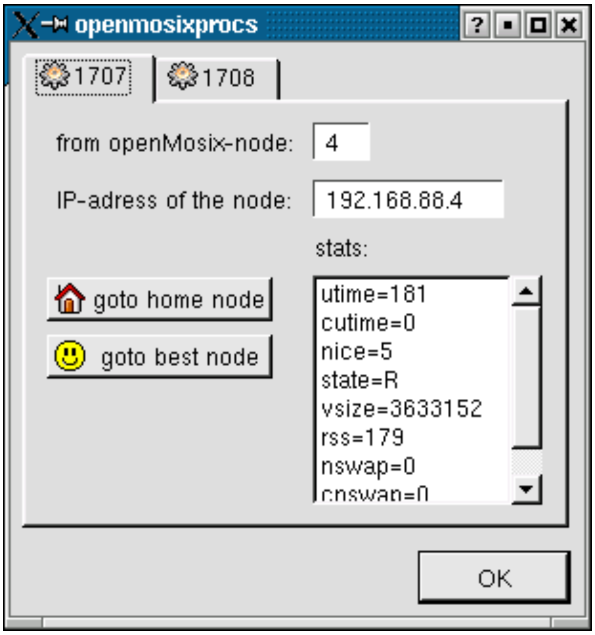
The openMosixview−migrator window displays all nodes in your openMosix−cluster. This window is for managing one process (with additional status−information). By double−clicking on an host from the list the process will migrate to this host. After a short moment the process−icon for the managed process will be green, which means it is running remote.

The "home"−button sends the process to its home node. With the "best"−button the process is send to the best available node in your cluster. This migration is influenced by the load, speed, CPU's and what openMosix "thinks" of each node. It maybe will migrate to the host with the most CPU's and/or the best speed. With the "kill"−button you can kill the process immediately.

To pause a program just click the "SIGSTOP"−button and to continue the "SIGCONT"−button. With the renice−slider below you can renice the current managed process (−20 means very fast, 0 normal and 20 very slow)

10.5.3. managing processes from remote

This dialog will pop up if the "manage procs from remote"−button beneath the process−box is clicked



from openMosix-node: 4

IP-adress of the node: 192.168.88.4

stats:

```
utime=181
cutime=0
nice=5
state=R
vsize=3633152
rss=179
nswap=0
cnswap=0
```

The TabView displays processes that are migrated to the local host. The procs are coming from other nodes in your cluster and currently computed on the host openMosixview is started on. Similar to the two buttons in the migrator−window the process is send home by the "goto home node"−button and send to the best available node by the "goto best node"−button.

10.6. openMosixcollector

The openMosixcollector is a daemon which should/could be started on one cluster−member. It logs the openMosix−load of each node to the directory /tmp/openmosixcollector/* These history log−files analyzed by the openMosixanalyzer (as described later) gives an nonstop overview of the load, memory and processes in your cluster. There is one main log−file called /tmp/openmosixcollector/cluster Additional to this there are additional files in this directory to which the data is written.

At startup the openMosixcollector writes its PID (process id) to /var/run/openMosixcollector.pid

The openMosixcollector−daemon restarts every 12 hours and saves the current history to /tmp/openmosixcollector[date]/* These backups are done automatically but you can also trigger this manual.

There is an option to write a checkpoint to the history. These checkpoints are graphically marked as a blue vertical line if you analyze the history log−files with the openMosixanalyzer. For example you can set a checkpoint when you start a job on your cluster and another one at the end..

Here is the explanation of the possible commandline−arguments:

```
openmosixcollector −d      //starts the collector as a daemon
openmosixcollector −k      //stops the collector
openmosixcollector −n      //writes a checkpoint to the history
openmosixcollector −r      //saves the current history and starts a new one
openmosixcollector         //print out a short help
```

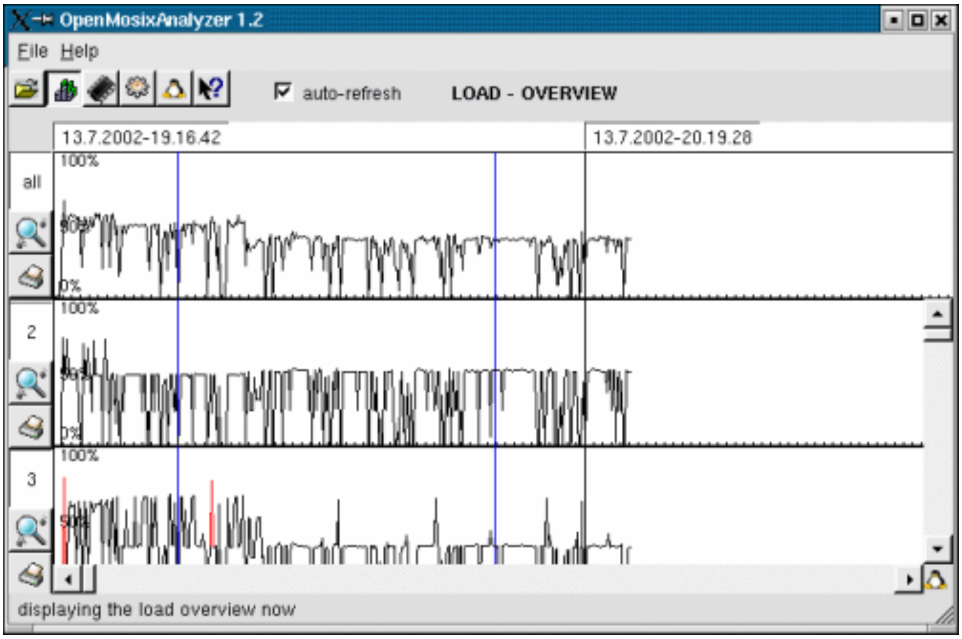
You can start this daemon with its init−script in /etc/init.d or /etc/rc.d/init.d. You just have to create a symbolic link to one of the runlevels for automatic startup.

How to analyze the created logfiles is described in the openMosixanalyzer−section.

10.7. openMosixanalyzer

10.7.1. the load−overview

This picture shows the graphical Load−overview in the openMosixanalyzer (Click to enlarge)



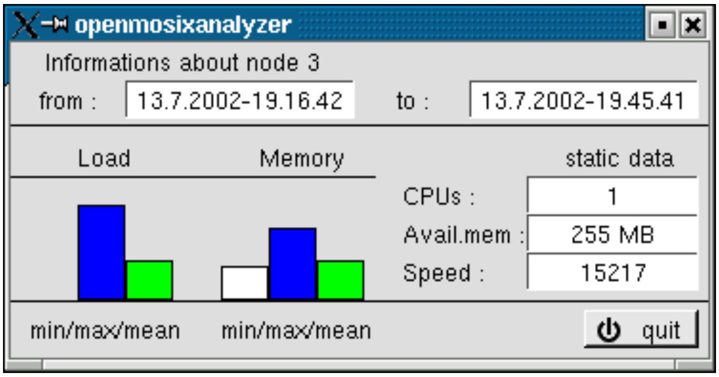
With the openMosixanalyzer you can have a non−stop openMosix−history of your cluster. The history log−files created by openMosixcollector are displayed in a graphically way so that you have a long−time overview what happened and happens on your cluster. The openMosixanalyzer can analyze the current "online" logfiles but you can also open older backups of your openMosixcollector history logs by the filemenu. The logfiles are placed in /tmp/openmosixcollector/* (the backups in /tmp/openmosixcollector[date]/*) and you have to open only the main history file "cluster" to take a look at older load−informations. (the [date] in the backup directories for the log−files is the date the history is saved) The start time is displayed on the top and you have a full−day view in the openMosixanalyzer (12 h).

If you are using the openMosixanalyzer for looking at "online"−logfiles (current history) you can enable the "refresh"−checkbox and the view will auto−refresh.

The load−lines are normally black. If the load increases to >75 the lines are drawn red. These values are openMosix−−informations. The openMosixanalyzer gets these informations from the files /proc/hpc/nodes/[openMosix ID]/*

The Find−out−button of each nodes calculates several useful statistic values. Clicking it will open a small new window in which you get the average load− and mem values and some more statically and dynamic informations about the specific node or the whole cluster.

10.7.2. statistical informations about a cluster−node



Informations about node 3

from : 13.7.2002-19.16.42 to : 13.7.2002-19.45.41

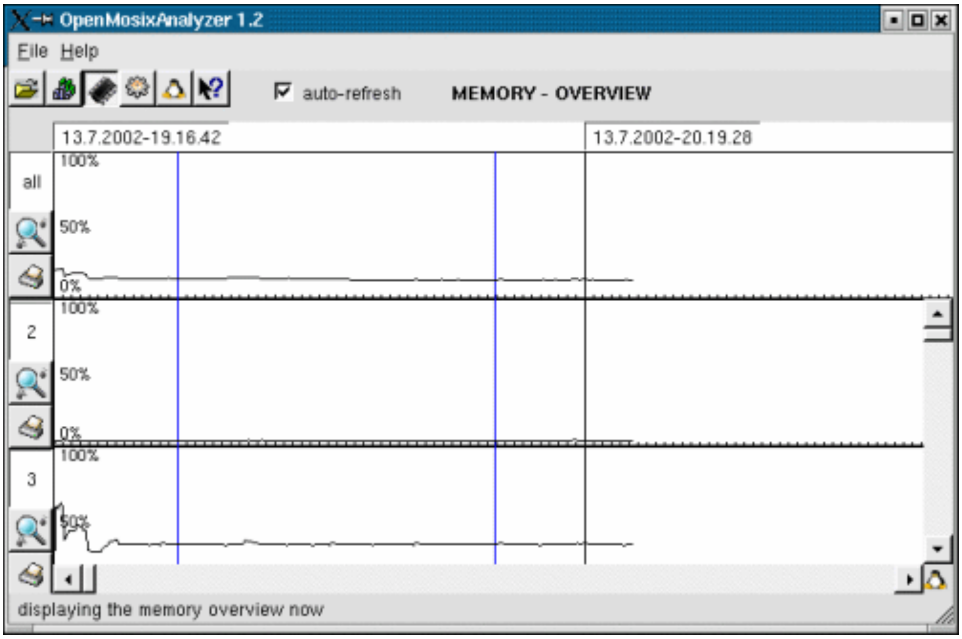
CPUs : 1

Avail.mem : 255 MB

Speed : 15217

If there are checkpoints written to the load−history by the openMosixcollector they are displayed as a vertical blue line. You now can compare the load values at a certain moment much easier.

10.7.3. the memory−overview



☑ auto-refresh

13.7.2002-19.16.42

13.7.2002-20.19.28

displaying the memory overview now

This picture shows the graphical Memory−overview in the openMosixanalyzer

With Memory−overview in the openMosixanalyzer you can have a non−stop memory history similar to the Load−overview. The history log−files created by openMosixcollector are displayed in a graphically way so that you have a long−time overview what happened and happens on your cluster. It analyze the current "online" logfiles but you can also open older backups of your openMosixcollector history logs by the filemenu.

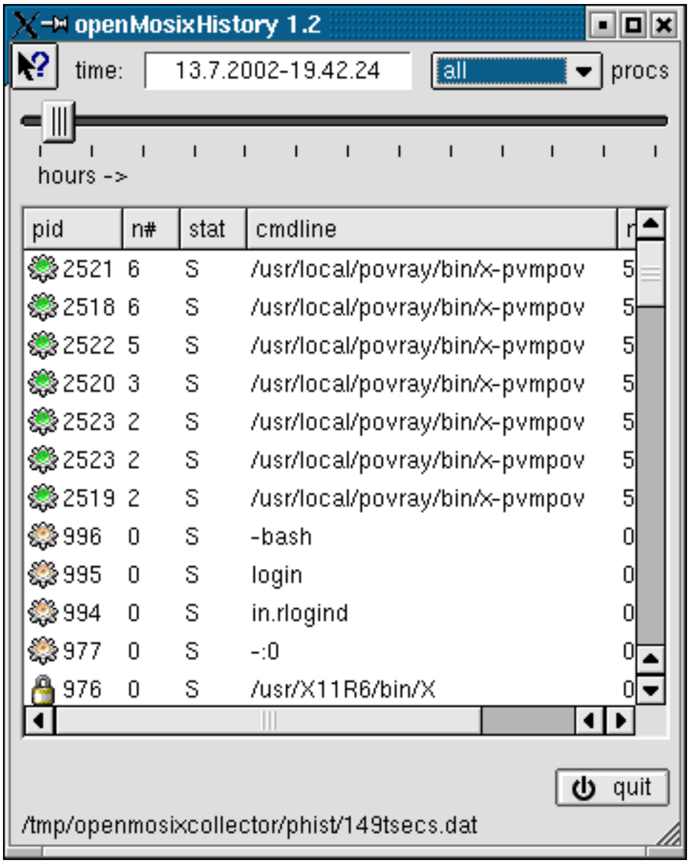
The displayed values are openMosix−informations. The openMosixanalyzer gets these informations from the files

```
/proc/hpc/nodes/[openMosix−ID]/mem.
/proc/hpc/nodes/[openMosix−ID]/rmem.
```

```
/proc/hpc/nodes/[openMosix−ID]/tmem.
```

If there are checkpoints written to the memory−history by the openMosixcollector they are displayed as a vertical blue line.

10.7.4. openMosixhistory



time: 13.7.2002-19.42.24 all procs

pid	n#	stat	cmdline	r
2521	6	S	/usr/local/povray/bin/x-pvmpov	5
2518	6	S	/usr/local/povray/bin/x-pvmpov	5
2522	5	S	/usr/local/povray/bin/x-pvmpov	5
2520	3	S	/usr/local/povray/bin/x-pvmpov	5
2523	2	S	/usr/local/povray/bin/x-pvmpov	5
2523	2	S	/usr/local/povray/bin/x-pvmpov	5
2519	2	S	/usr/local/povray/bin/x-pvmpov	5
996	0	S	-bash	0
995	0	S	login	0
994	0	S	in.rlogind	0
977	0	S	-:0	0
976	0	S	/usr/X11R6/bin/X	0

/tmp/openmosixcollector/phist/149tsecs.dat

displays the processlist from the past

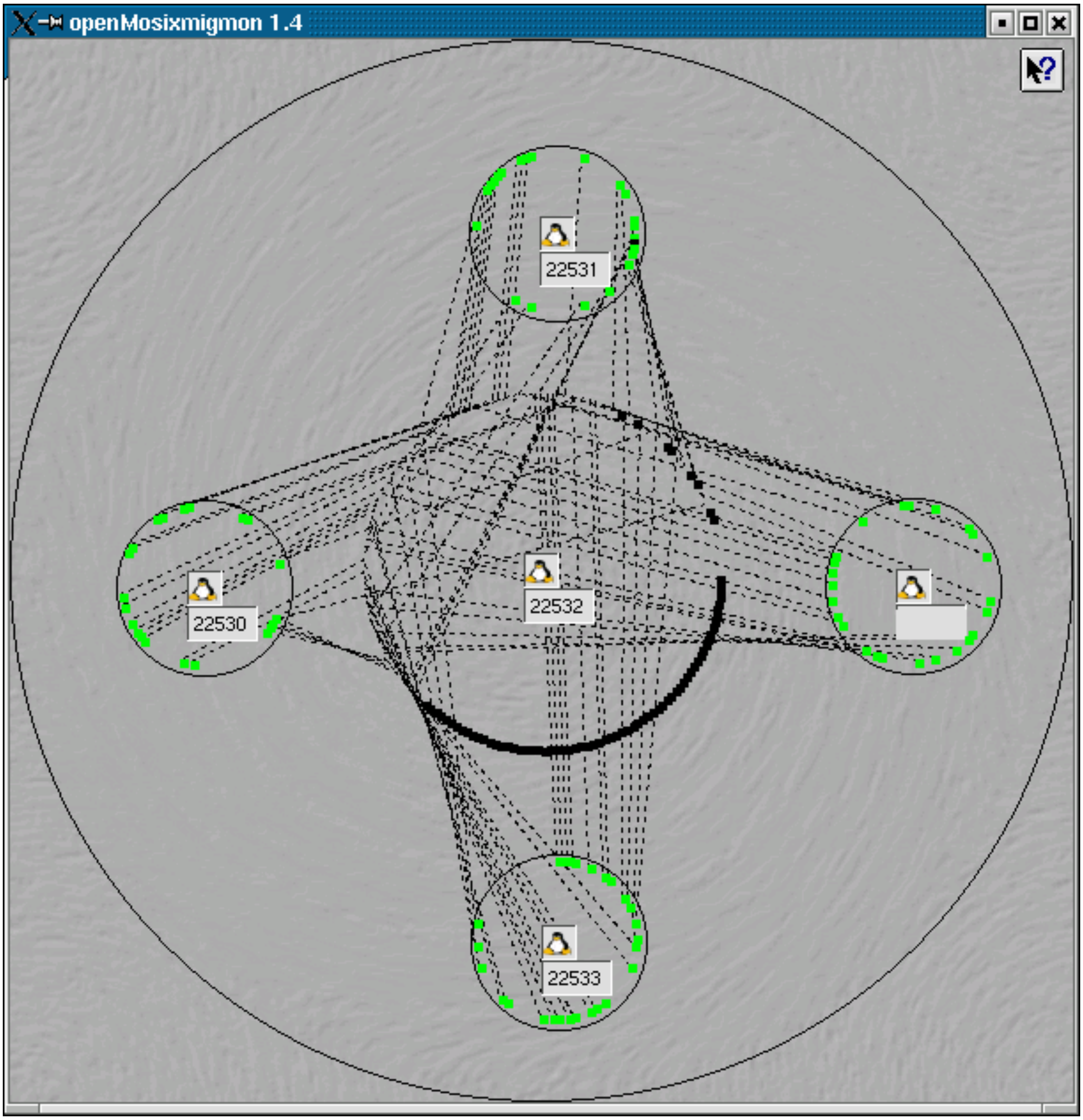
openMosixhistory gives a detailed overview which process was running on which node. The openMosixcollector saves the processlist from the host the collector was started on and you can browse this log−data with openMosixhistory. You can easy change the browsing time in openMosixhistory by the time−slider.

openMosixhistory can analyze the current "online" logfiles but you can also open older backups of your openMosixcollector history logs by the filemenu.

The logfiles are placed in /tmp/openmosixcollector/* (the backups in /tmp/openmosixcollector[date]/*) and you have to open only the main history file "cluster" to take a look at older load−informations. (the [date] in the backup directories for the log−files is the date the history is saved) The start time is displayed on the top/left and you have a 12 hour view in openMosixhistory.

10.8. openMosixmigmon

10.8.1. General



The openMosixmigmon is a monitor for migrations in your openMosix−cluster. It displays all your nodes as little penguins sitting in a circle.

−> *nodes−circle.*

The main penguin is the node on which openMosixmigmon runs and around this node it shows its processes also in a circle of small black squares.

−> *main process−circle*

If a process migrates to one of the nodes the node gets an own process−circle and the process moved from the main process−circle to the remote process−circle. Then the process is marked green and draws a line from its origin to its remote location to visualize the migration.

10.8.2. Tooltips:

If you hold your mouse above a process it will show you its PID and commandline in a small tooltip−window.

10.8.3. Drag'n Drop!

The openMosixmigmon is fully Drag'n Drop enabled. You can grab (drag) any process and drop them to any of your nodes (those penguins) and the process will move there. If you double−click a process on a remote node it will be send home immediately.

10.9. openmosixview FAQ

10.9.1. I cannot compile openMosixview on my system?

At first QT >= 2.3.x is required. The QTDIR −environment variable has to be set to your QT−installation directories like it is well described in the INSTALL− file. In versions < 0.6 you can do a "make clean" and delete the two files: /openmosixview/Makefile /openmosixview/config.cache and try to compile again because i alway left the binary− and object−files in older versions. If you have any other problems post them to the openMosixview−mailinglist (or directly to me).

10.9.2. Can I use openMosixview with SSH?

Yes, until version 0.7 there is a built−in SSH−support. You have to be able to ssh to each node in your cluster without password (just like the same with using RSH this is required)

10.9.3. I started openMosixview but only the splash−screen appears. What is wrong?

Do not fork openMosixview in the background with & (e.g. openMosixview &). Maybe you cannot rsh/ssh (depends on what you want to use) as user root without password to each node? Try "rsh hostname" as root. You should not been promped for a password but soon get a login shell. (If you use SSH try "ssh hostname" as root.) You have to be root on the cluster because that is the only way the administrative commands executed by openMosixview requires root−privileges. openMosixview uses "rsh" as the default! If you only have "ssh" installed on your cluster edit (or create) the file /root/.openMosixview and put "1111" in it. This is the main−configuration file for openMosixview and the last "1" stands for "use ssh instead of rsh". This will cause openMosixview to use "ssh" even for the first start.

10.9.4. The openMosixviewprocs/mosixview_client is not working for me!

The openMosixview−client is executed per rsh (or ssh which you can configer whith a checkbox) on the remote host. It has to be installed in /usr/bin/ on each node. If you use RSH try: "xhost +hostname" "rsh hostname /usr/bin/openMosixview_client −display your_local_host_name:0.0" or if you use SSH try: "xhost +hostname" "ssh hostname /usr/bin/openMosixview_client −display your_local_host_name:0.0" If this works it will work in openMosixview too. openMosixview crashes with "segmentation fault"! Maybe you still use an old version of openMosixview/Mosixview ? in the mosix.map−parser (which is completly removed in openMosixview !!) (the versions openMosixview 1.2 and Mosixview > 1.0 are stable)

10.9.5. Why are the buttons in the openMosixview−configuration dialog not preselected?

(automigration on/off, blocking on/off......) I want them to be preselected too. The problem is to get the information of node. You have to login to each cluster−node because these information are not cluster−wide (to my mind). The status of each node is stored in the /proc/hpc/admin directory of each node. Everybody who knows a good way to get these information easy is invited to mail me.

10.10. openMosixview + ssh:

(this HowTo is for SSH2) You can read the reasons why you should use SSH instead of RSH everyday on the newspaper when another script−kiddy hacked into an insecure system/network. So SSH is a good decision at all.

```
freedom x security = constant    (from a security newsgroup)
```

That is why it is a bit tricky to configure SSH. SSH is secure even if you use it to login without being prompted for a password. Here is a (one) way to configure it.

At first a running secure−shell daemon on the remote site is required. If it is not already installed install it! (rpm −i [sshd_rpm_packeage_from_your_linux_distribution_cd]) If it is not already running start it with:

```
/etc/init.d/ssh start
```

Now you have to generate a keypair for SSH on your local computer whith ssh−keygen.

```
ssh−keygen
```

You will be prompt for a passphrase for that keypair. The passphrase normally is longer than a password and may be a whole sentence. The keypair is encrypted with that passphrase and saved in

```
/root/.ssh/identity    //your private key
and
/root/.ssh/identity.pub     //your public key
```

Do NOT give your private−key to anybody!!! Now copy the whole content of /root/.ssh/identity.pub (your public−key which should be one long line) into /root/.ssh/authorized_keys on the remote host. (also copy the content of /root/.ssh/identity.pub to your local /root/.ssh/authorized_keys like you did it with the remote−node because openMosixview needed password−less login to the local−node too!)

If you ssh to this remote host now you will be prompted for the passphrase of your public−key. Giving the right passphrase should give you a login.

What is the advantage right now??? The passphrase is normally a lot longer than a password!

The advantage you can get using the ssh−agent. It manages the passphrase during ssh login.

```
ssh−agent
```

The ssh−agent is started now and gives you two environment−variables you should set (if not set already). Type:

```
echo $SSH_AUTH_SOCK
and
echo $SSH_AGENT_PID
```

to see if they are exported to your shell right now. If not just cut and paste from your terminal. e.g. for the bash−shell:

```
SSH_AUTH_SOCK=/tmp/ssh−XXYqbMRe/agent.1065
export SSH_AUTH_SOCK
SSH_AGENT_PID=1066
export SSH_AGENT_PID
```

example for the csh−shell:

```
setenv SSH_AUTH_SOCK /tmp/ssh−XXYqbMRe/agent.1065
setenv SSH_AGENT_PID 1066
```

With these variables the remote−sshd−daemon can connect your local ssh−agent by using the socket−file in /tmp (in this example /tmp/ssh−XXYqbMRe/agent.1065). The ssh−agent can now give the passphrase to the remote host by using this socket (it is of course an encrypted transfer)!

You just have to add your public−key to the ssh−agent with the ssh−add command.

```
ssh−add
```

Now you should be able to login using ssh to the remote host without being prompted for a passwod!

You could (should) add the ssh−agent and ssh−add commands in your login−profile e.g.

```
eval `ssh−agent`
 ssh−add
```

Now it is started when you login on your local workstation. You have done it! I wish you secure logins now.

openMosixview There is a menu−entry which toggles using rsh/ssh with openMosixview. Just enable this and you can use openMosixview even in insecure network−environments. You should also save this configuration (the possibility for saveing the current config in openMosixview was added in the 0.7 version) because it gets initial data from the slave using rsh or ssh (just like you configured).

If you choose a service wich is not installed properly openMosixview will not work! (e.g. if you cannot rsh to a slave without being prompted for a password you cannot use openMosixview with RSH; if you cannot ssh to a slave without being prompted for a password you cannot use openMosixview with SSH)

Chapter 11. Other openMosix related Programs

11.1. Introduction

There are a couple of different applications available to monitor and admin openMosix, we give a short overview of them in this chapter, we won't really go in detail.

11.2. openMosixView

openMosixview is the most used and the best known applet for openMosix administration, you can read more about it in the openMosix adminstration chapter.

11.3. openMosixapplet

The openMosixApplet lets you watch the realtime load of your openMosix cluster. It consists of a local daemon which listens for connections by applets. The applet uses chart2D to provide a good−lookin' feeling.

11.4. wmonload

wmomload is a simple, but handy and small dockapp for overviewing the load of cluster nodes in a small openMosix−based cluster.

11.5. openMosixWebView

openMosixWebView − Produces web charts for monitoring an openMosix cluster. openMosixWebView is a PHP script for monitoring an openMosix cluster via the WEB. It uses openMosixview's openMosixCollector logs. Download now the last release *openmosixwebview−0.2.12.tar.gz (16 Feb 2003)*

See openMosixWebView screenshots and running :−) Released under the GNU General Public License (GPL). See README and FAQ files.

Chapter 12. Common Problems

12.1. Introduction

Although most of the issues in this chapter could be a part of the FAQ. There where the FAQ will give a short "how to solve" answer, we have taken a closer look at them and explained why they are problems and how to solve them.

12.2. My processes won't migrate

Help process XYZ doesn't migrate. Moshe Bar explains below why some processes migrate and why some don't. But before that you can always look in /proc/$pid/, there often is a *cantmove* file which will tell you why a certain process can't migrate.

Processes can also be locked. You can check if a process is locked with:

```
cat /proc/$PID/lock
```

where $PID is the processid of the process in question.

Now listen to what Moshe himself has to say about this topic.

Often people have the same kernel but on a different distribution, say a mixed environment of RedHat and Debian ,rc scripts from different distros tend to start openmosix differently. Some implementations completely modify /etc/inittab to start all daemons (and their children) with

```
mosrun −h
```

. So that they won't migrate. Therefore all these processes have a 1 in /proc/pid/lock when you start. You can force them to migrate by writing a 0 to this file .

Ok, this simple program should always migrate if launched more times than number of local CPUs. So for a 2−way SMP system, starting this program 3 times will start migration if the other nodes in the cluster have at least the same speed like the local ones:

```
int main() {
    unsigned int i;
    while (1) {
        i++;
    }
    return 0;
}
```

On a Pentium 800Mhz CPU it takes quite a while to overflow.

This sample program with content like this will never migrate:

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
```

```
...

key_t key; /* key to be passed to shmget() */
int shmflg; /* shmflg to be passed to shmget() */
int shmid; /* return value from shmget() */
int size; /* size to be passed to shmget() */

...

key = ...
size = ...
shmflg) = ...

if ((shmid = shmget (key, size, shmflg)) == -1) {
   perror("shmget: shmget failed"); exit(1); } else {
   (void) fprintf(stderr, "shmget: shmget returned %d\n", shmid);
   exit(0);
  }
...
```

Program using pipes like this do migrate nicely:

```
int pdes[2];

pipe(pdes);
if ( fork() == 0 )
  { /* child */
                                 close(pdes[1]); /* not required */
                                 read( pdes[0]); /* read from parent */
                                 .....
                 }
else
                 { close(pdes[0]); /* not required */
                                 write( pdes[1]); /* write to child */
                                 .....
                 }
```

MODIFIED Programs using pthreads since version 2.4.17 don't migrate, however they don't segfault anymore.

```
//
// Very simple program demonstrating the use of threads.
//
// Command-line argument is P (number of threads).
//
// Each thread writes "hello" message to standard output, with
//   no attempt to synchronize.  Output will likely be garbled.
//
#include <iostream>
#include <cstdlib>              // has exit(), etc.
#include <unistd.h>             // has usleep()
#include <pthread.h>            // has pthread_ routines

// declaration for function to be executed by each thread
void * printHello(void * threadArg);

// ---- Main program -------------------------------------------------

int main(int argc, char* argv[]) {
  if (argc < 2) {
    cerr << "Usage:  " << argv[0] << " numThreads\n";
    exit(EXIT_FAILURE);
  }
  int P = atoi(argv[1]);

  // Set up IDs for threads (need a separate variable for each
  //   since they're shared among threads).
  int * threadIDs = new int[P];
  for (int i = 0; i < P; ++i)
    threadIDs[i] = i;

  // Start P new threads, each with different ID.
  pthread_t * threads = new pthread_t[P];
  for (int i = 0; i < P; ++i)
    pthread_create(&threads[i], NULL, printHello,
                   (void *) &threadIDs[i]);

  // Wait for all threads to complete.
  for (int i = 0; i < P; ++i)
    pthread_join(threads[i], NULL);

  // Clean up and exit.
  delete [] threadIDs;
  delete [] threads;
  cout << "That's all, folks!\n";
  return EXIT_SUCCESS;
}

// ---- Code to be executed by each thread ---------------------------

// pre:  *threadArg is an integer "thread ID".
// post:  "hello" message printed to standard output.
//        return value is null pointer.
void * printHello(void * threadArg) {
  int * myID = (int *) threadArg;
  cout << "hello, world, ";
  // pointless pause, included to make the effects of
  //   synchronization (or lack thereof) more obvious
  usleep(10);
  cout << "from thread " << *myID << endl;
  pthread_exit((void* ) NULL);
}
```

Programs using all kinds of file descriptors, including sockets do migrate (sockets are not migrated with the process however, files are migrated if using oMFS/DFSA)

(all above code is by Moshe as Moshe Bar or by Moshe as CTO of Qlusters, Inc.)

Please also refer to the man pages of openMosix , they also give an adequate explanation why processes don't migrate.

If for some reason your processes stay locked while they shouldn't. You can try to allow locked processes to migrate by simply putting

```
# tell shells to allow subprocs to migrate to other nodes
echo 0 > /proc/self/lock
```

into "/etc/profile" Warning : this fix will allow *all* process to migrate not just the ones you want. To only allow specific process to migrate use 'mosrun −l' to unlock only the desired process.

12.3. I don't see all my nodes

First of all , are you using the same kernel version on each machine ? The 'same−kernel' refers to the version. You can build different kernel images of the same source version to meet the hardware/software needs of a given node. However you wil need toe make sure that when you install openMosix on your cluster, all your machines should have the openmosix−x.x.x−y kernel installed, in contrast to having one machine running openmosix−x.x.z−x, another running openmosix−x.x.x−y, another running openmosix x.x.x−z, and so on and so forth

When you run mosmon, press t to see the total of machines running. Does it warn you that mosix is not running?

If yes, then make sure your machine's ip is included in /etc/mosix.map (don't use 127.0.0.1 − if your machine's ip is such, then you probably have problems with your dhcp server/nameserver). If it does not tell you that mosix is not running, see what machines show up. Do you see only your machine?

If yes, then your machine is most likely running a firewall and is not letting openmosix through.

If not, then the problem is most likely with the machine that doesn't show up. Also: Do you have two nic cards on a node? then you have to edit the /etc/hosts file to have a line that has the following format

```
non−cluster_ip  cluster−hostname.cluster−domain cluster−hostname
```

You might also need to set up a routing table, which is a whole different subject.

Maybe you used different kernel−parameters on each machine? Especially if you use the 'Support clusters with a complex network topology' option you should take care that you use the same value for the also appearing option 'Maximum network−topology complexity support' on each machine.

12.4. I often get errors: No such process

I often get the error

```
bash: child setpgid (4061 to 4061): No such process
```

what does this mean ?

The above line meas that the shell you were using has acutallly migrated to another node ? This printout from bash is caused by a bug in old version of openmosix, but a fix has been commited. (Muli Ben−Yehuda mulix@actcom.co.il)

12.5. DFSA ? MFS ?

People often get confused about what exactly MFS and DFSA are. As discussed before in the howto MFS is the feature of openMosix that enables you access to remote filesystems as if those filesystems were locally mounted. They are mostly mounted on /mfs . A common misunderstanding is that you need MFS in order to have openMosix working, this is not true, however it can make things easier.

With DFSA enabled, system calls will be executed on the remote node withouth migrating the process back to it's home node. This behaviour (direct filesystem access) causes processes migratiing to the data and not the other way around (which is common). If DFSA is not enabled MfS is "just" a non−caching network−filesystem.

Very generally speaking, if you don't have DFSA turned on, each and every I/O will go to the home node for execution. With DFSA turned on, if the file happens to be residing on the node where the process finds itself then the I/O will happen locally.

A very common error is that people mix kernels with DFSA enabled and disabled. So one has to have a way to find out wether DFSA is actually enabled. This information can be obtained by typing

```
cat /proc/hpc/admin/version
```

12.6. Python Troubles

Some people have reported problems with Python, closer research showed that these problems were not with openMosix but rather with glibc issues, however it seemed that issues manifested themselves especially in openMosix.

One user solved the problem by removig /lib/i686/lib* on his machine and let the applications link dynamically against /lib/libpthread (and other) However bugfixes in newer glibc versions combined with more recent openMosix version seem to have solved these problems.

Chapter 13. Hints and Tips

13.1. Locked Processes

If for some reason you find your processes are always locked in your home node and you can't find the reason, you can put the following lines into your ~/.profile as a stop−gap measure to automatically enable migration:

```
        if [ −x /proc/$$/lock ]; then
           echo 0 > /proc/$$/lock
        fi
```

However, you should make an effort to find out what the problems is

13.2. Choosing your processes

You will probably want to test your setup before deciding which programs you want to enable migration for. For example, if you are running KDE2 on a slow machine and have a significantly faster machine has part of your Mosix cluster, you might find resource−hungry programs like kmail are migrated out. This is not a bad thing as such, however, it can lead to a brief moment when your writing is not displayed on the screen immediately.

13.3. Java and openMosix

Green Threads JVM's, allow for migration because each Java thread is a separate process. Threads other than Java green thread JVM's cannot be migrated by Linux, so openMosix cannot migrate programs that use them.

If you have the source so your Java application you might be able to compile the application native. In this case you might be able to migrate your applications to another node.

Gian Paolo Ghilardi wrote a paper titled Consideration on OpenMosix it deals amongst other topics with Java an dopenMosix. *http://www.democritos.it/events/openMosix/papers/crema.pdf*

13.4. openMosix and Hyperthreading

Basically openMosix performance increases with the current Linux scheduler when Hypethreading is disabled. You can do this by either entering 'noht' as a boot option or disabling HT in the bios.

For those who are still wondering what hypetrheading is : *Intel explains it*

13.5. openMosix and Firewalls

People often have questions regarding openMosix and firewalls. Amit helped me out on this matter:

```
from hpc/comm.c:

#define MIG_DAEMON_PORT         0x3412
#define INFO_DAEMON_PORT        0x3415
```

converted to decimal, they are: 4660 and 5428 resp. (convert 1234, and not 3412, 'cos of the network−host byte ordering conversions.. Read the IP/TCP/UDP RFCs for info)

the mig_daemon port is a tcp port, the info_daemon port is udp. Hence tcp/4660 and udp/5428, Matt also mentions tcp/723 somewhere.

Chapter 14. (stress)Testing your openMosix installation

14.1. A small Test Script

The fastest way to test your openMosix cluster is by creating a small script with the following content.

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

I've saved it as test_mosix, now when I want to see If everything works I start op mosmon and I launch this script for a zillion times as in

```
for i in `ls /etc/` ; do ./test_mosix ; done
```

Now watch openMosix kicking in after a few seconds ...

Just pkill awk on the home node to kill'em all ;)

14.2. Perl Proggie by Charles Nadeau

Perl program to test an openMosix Cluster.

Here is a is quick program I wrote to test an openMosix cluster. This is taken from a posting I made to the openMosix−devel mailing list on March 6th, 2002: "Charles wrote this little program (in Perl) to stress test his home cluster (3 P200MMX and a P166). It is a program simulating different sets of stocks in a portfolio for a given period of time. The code is well documented and it should be easy to add/remove stocks and change the average monthly yield and standard deviation for each stock. Since the problem of portfolio optimization cannot be solved analytically, it simulate a lot of portfolios and report the results at the end. Please note that this program does not take stock correlation into account. It is not finished yet but it's a good start. I plan to add more code at the end of the program to improve the reporting format of the data (generating SVG graph on the fly). But the simulation part works very well. In order to take advantage of the parallelism offered by openMosix, it uses the Perl module Parallel::?ForkManager (from CPAN) to span threads that openMosix can then assign to all the machines of the cluster (it also require another module for the statistical calculations, don't forget to install both, I provide the URLs in the comments of the code). Take a look at it and tell me what you think. Cheers!"

```
#! /usr/bin/perl −w

# this mill unlock this process and all tis childs
sub unlock {
open (OUTFILE,">/proc/self/lock") ||
die "Could not unlock myself!\n";
print OUTFILE "0";
}

unlock;

# this will count the nodes
sub cpucount {
 $CLUSTERDIR="/proc/hpc/nodes/";
 $howmany=0;
 opendir($nodes, $CLUSTERDIR);
 while(readdir($nodes)) {
  $howmany++;
 }

 $howmany−−;
 $howmany−−;
 closedir ($nodes);
 return $howmany;
}

my $processes=cpucount;
$processes=$processes*3;

print("starting $processes processes\n");

#Portfolio.pl, version 0.1
#Perl program that simulate a portfolios for various stock composition for a given period of tim
#We run various scenarios to find the mix of assets that give the best performance/risk ratio
#This method is base on the book "The intelligent asset allocator" by William Bernstein
#Can be used to test an OpenMosix cluster
#This program is licensed under GPL
#Author: Charles−E. Nadeau Ph.D., (c) 2002
#E−mail address: charlesnadeau AT hotmail DOT com

use Parallel::ForkManager; #We use a module to parallelize the calculation

#Available at http://theoryx5.uwinnipeg.ca/mod_perl/cpan−search?filetype=%20distribution%20name%2

use Statistics::Descriptive::Discrete; #A module providing statistical values

#Available at http://theoryx5.uwinnipeg.ca/mod_perl/cpan−search?new=Search;filetype=%20distributi

srand; #We initialize the random number generator

#Initializing constant
$NumberOfSimulation=$processes;  #Number of simulation to run
$NumberOfMonth=100000; #Number of month for wich to run the simulation
$NumberOfStock=6; #Number of different stocks in the simulation

#Portfolio to simulate
#TODO: Read the stock details from a file
$Stock[0][0]="BRKB"; #Stock ticker
$Stock[0][1]=0.01469184; #Stock average monthly return
$Stock[0][2]=0.071724934; #Stock average monthly standard deviation
$Stock[1][0]="TEST "; #Stock ticker
$Stock[1][1]=−0.01519; #Stock average monthly return
$Stock[1][2]=0.063773903; #Stock average monthly standard deviation
$Stock[2][0]="SPDR"; #Stock ticker
$Stock[2][1]=0.008922718; #Stock average monthly return
$Stock[2][2]=0.041688404; #Stock average monthly standard deviation
$Stock[3][0]="BRKB"; #Stock ticker
$Stock[3][1]=0.01469184; #Stock average monthly return
$Stock[3][2]=0.071724934; #Stock average monthly standard deviation
$Stock[4][0]="TEST "; #Stock ticker
$Stock[4][1]=−0.01519; #Stock average monthly return
$Stock[4][2]=0.063773903; #Stock average monthly standard deviation
$Stock[5][0]="SPDR"; #Stock ticker
$Stock[5][1]=0.008922718; #Stock average monthly return
$Stock[5][2]=0.041688404; #Stock average monthly standard deviation
```

```
my $pm = new Parallel::ForkManager($NumberOfSimulation); #Specify the number of threads to span

$pm->run_on_start(
  sub { my ($pid,$ident)=@_;
  print "started, pid: $pid\n";
  }
);



#We initialize the array that will contain the results
@Results=();
for $i (0..$NumberOfSimulation-1){
        for $j (0..$NumberOfStock+3){
                $Results[$i][$j]=0.0; #Equal to 0.0 to start
        }
}

for $i (0..$NumberOfSimulation-1){ #Loop on the number of simulation to run
        $Results[$i][0]=$i; #The first column of each line is the number of the simulation
        $pm->start and next; #Start the thread

                $TotalRatio=1; #The sum of the proprtion of each stock

                #Here we calculate the portion of each investment in the portfolio for a given si
                for $j (0..$NumberOfStock-2){ #We loop on all stock until the second to last one
                        #TODO: Replace rand by something from Math::TrulyRandom
                        $Ratio[$j]=rand($TotalRatio);
                        $Results[$i][$j+1]=$Ratio[$j]; #We store the ratio associated to this sto
                        $TotalRatio=$TotalRatio-$Ratio[$j];
                }
                $Ratio[$NumberOfStock-1]=$TotalRatio; #In order to have a total of the ratios equ
                $Results[$i][$NumberOfStock]=$Ratio[$NumberOfStock-1]; #We store the ratio associ

                $InvestmentValue=1; #Initially the investment value is 1 time the initial capital
                my $stats=new Statistics::Descriptive::Discrete; #We initialize the module used t

                for $j (1..$NumberOfMonth){ #Loop on the number of months

                        $MonthlyGrowth[$j]=0.0; #By how much did we grow this month. Initially, 

                        #We loop on each stock to find its monthly contribution to the yield
                        for $k (0..$NumberOfStock-1){

                        $MonthlyGrowth[$j]=$MonthlyGrowth[$j]+($Ratio[$k]*((gaussian_rand()*$Sto
                        }

                        $stats->add_data($MonthlyGrowth[$j]); #Add the yield for this month so we
                        $InvestmentValue=$InvestmentValue*(1+$MonthlyGrowth[$j]); #Value of the I
                }
                $Results[$i][$NumberOfStock+1]=$stats->mean(); #Calculate the average monthly gro
                $Results[$i][$NumberOfStock+2]=$stats->standard_deviation(); #Calculate the stan

        $pm->finish; #Finish the thread
}
$pm->wait_all_children; #We wait until all threads are finished

#Printing the results
print "Simulation ";
for $j (0..$NumberOfStock-1){
        print "Ratio$Stock[$j][0] ";
```

```
}
print "  Mean StdDev YieldRatio\n";
for $i (0..$NumberOfSimulation-1){
        printf "%10d ", $Results[$i][0];
        for $j (1..$NumberOfStock){
                printf "   %6.2f ",$Results[$i][$j];
        }

        if($Results[$i][$NumberOfStock+2]!=0) {
                printf "%5.4f %5.4f    %5.4f\n", $Results[$i][$NumberOfStock+1], $Results[$i][$Nu

        } else {
                printf "%5.4f %5.4f    %5.4f\n", $Results[$i][$NumberOfStock+1], $Results[$i][$Nu

        }
}



#Subroutines

#Subroutine to generate two numbers normally distributed
#From "The Perl Cookbook", recipe 2.10
sub gaussian_rand {
        my ($u1, $u2);
        my $w;
        my ($g1, $g2);

        do {
                $u1=2*rand()-1;
                $u2=2*rand()-1;
                $w=$u1*$u1+$u2*$u2;
        } while ($w>=1 || $w==0); #There was an error in the recipe, I corrected it here

        $w=sqrt(-2*log($w)/$w);
        $g2=$u1*$w;
        $g1=$u2*$w;
        return wantarray ? ($g1,$g2) : $g1;

}
```

14.3. the openMosix stress−test

by Matt Rechenburg

14.3.1. General description

This stress test is made to test an openMosix cluster + kernel. It will perform several application + kernel tests for checking the stability and other features of openMosix (e.g. process migration, mfs, ...). During this test the cluster will be mostly loaded so you should stop other running applications before starting it. When it finished it generates a fully detailed report about each component which was tested.

14.3.2. Detailed description

The openMosix stress−test runs several programs to check the functionality of the whole system. In the following part you will find a description of each test−application:

- *distkeygen*: This applicaton is used to generate 4000 RSA key pairs with 1024 bits of key length. It is destributed into as many processes as processors in your openMosix cluster via fork.

  Requires : gcc compiler and OpenSSL library Copyright (C) 2001 Ying−Hung Chen (GPL) http://www.yingternet.com/mosix
- *portfolio* 'portfolio' is a perl program that simulate a portfolios for various stock composition for a given period of time. This method is base on the book "The intelligent asset allocator" by William Bernstein.

  This program is licensed under GPL Author: Charles−E. Nadeau Ph.D., (c) 2002 E−mail address: charlesnadeau AT hotmail DOT com
- *eatmem* : Simply calculates sin+sqrt from a value 1000000 times and outputs it outputs the loop count to a file (which will grow a lot) This tests is started as many times at once as many processors you have in your openMosix cluster automatically.
- *forkit*: The 'forkit' test is similar to the 'eatmem' test but uses fork to create multiple process (3*[processors_in_your_openMosix_cluster]) expect it does not write to files.
- *mfstest* This will create a 10MB file and copy it to all nodes back and forth. It is for checking the oMFS capabilities.
- *kernel syscall test*: The Linux Test Project is a joint project with SGI, IBM, OSDL, and Bull with a goal to deliver test suites to the open source community that validate the reliability, robustness, and stability of Linux. The Linux Test Project is a collection of tools for testing the Linux kernel and related features. The goal is to improve the Linux kernel by bring test automation to the kernel testing effort. Interested open source contributors are encouraged to join the project. For more informations visit : http://ltp.sf.net
- *moving*: The 'moving.sh' will move the 'start_openMosix_test.sh' around each node in your openMosix cluster while running the stress−test itself. So 'start_openMosix_test.sh' will migrate every minute to another node during the test−run. Dependent on how long the test will run on your cluster it will be migrated 20−40 times.

14.3.3. Installing the strestest suite

First of all download the rpm or source package from *http://www.openmosixview.com/omtest/*

- *using the source package :*

  Unzip and untar the openMosix stress−test with the following commands in e.g. /usr/local:

  ```
            gunzip omtest.tar.gz
            tar −xvf omtest.gz
  ```

  Then 'cd' into the /usr/local/omtest directory and execute:

  ```
            ./compile_tests.sh
  ```
  This will install the required perl modules and compile the test−programs. The installation of these modules requires root−privileges. Later you can run the openMosix stress−test also as a regular user.

(you maybe have to delete old temporary files from root's test−runs in /tmp because you won't have the permission to overwrite it as regular user) You are ready to run the test now with the command:

```
          ./start_openMosix_test.sh
```

- *using the RPM−package*

  There are some requirements to be met when installing the omtest.rpm, you will need e.g expect and compat−libstdc++−7.3−2.96.110 (If you are on RH 8.0) Just install the omtest.rpm with the following command:

  ```
            rpm −ihv omtest.rpm
  ```

  Now you can start the openMosix stress test with the command:

  ```
            start_openMosix_test.sh
  ```

  (The RPM−package will be installed in /usr/local/omtest) (Please not that the RPM wil allso install some perl modules).

14.3.4. Running the tests

```
[root@dhcp51 omtest]# ./start_openMosix_test.sh


starting the openMosix stress test now!

the results will be saved in : /tmp/openMosix−stress−test−report−03/16/2003−11:27:02.txt

oMFS is not mounted at /mfs! oMFS−test will be disabled.
Please mount oMFS before running this openMosix−test
You will find instructions how to configure and use oMFS at:
http://howto.ipng.be/openMosix−HOWTO/x222.htm#AEN243


(return to continue, ctrl−c for cancel)
```

IV. Running Applications on openMosix

Table of Contents

Chapter 15. Improving Compiling Performance

15.1. Introduction

This Section is a Work in Progress

Lots of people try to use openMosix as a kind of compile farm, ofthen they come back very disappointed. This chapter of the howto will try to explain in which cases your compilations will benefit from openMosix and how to improve your successlevel.

First of all you have to remember 1 thing. openMosix will not migrate *all* processes you start on your cluster, only the ones that will benefit from migration to another node. For compiling this means that a process has to last long enough. Kernel compiles typically consist of numerous short compiles, each of them not being long enough to acutally migrate.

Chapter 16. Imaging with openMosix

16.1. Introduction

This Section is a Work in Progress

Computer Graphics have always been applications that required a lot of CPU power, this hasn't changed. Within this chapter I wil demonstrate with some practical examples how Computer Graphics can benefit from openMosix.

16.2. Povray

The Persistence of Vision Raytracer is a high−quality, totally free tool for creating stunning three−dimensional graphics.

Ray−tracing is a rendering technique that calculates an image of a scene by simulating the way rays of light travel in the real world. However it does its job backwards. In the real world, rays of light are emitted from a light source and illuminate objects. The light reflects off of the objects or passes through transparent objects. This reflected light hits our eyes or perhaps a camera lens. Because the vast majority of rays never hit an observer, it would take forever to trace a scene.

These kind of applications can be easily made parrallel by using pvmpovray. Pvmpovray expects to working on a Beowulf style cluster and spread it's load to other nodes using pvm. The openMosix way of doing this is the same, however we just do this on 1 machine and have openMosix do the load spreading work fo you !

A GREAT Howto on PVM Povray will show you how to setup PVMPovray. Below is a small summary.

```
$ cd pvmpov3_1g_2
$ tar xfz ../povuni_s.tgz
$ tar xfz ../povuni_d.tgz
$ ./inst−pvm
Trying to apply the patch.
Searching for rejected files
$
```

Now compile with aimk (which is a wrapper script provided by the pvm rpm , buth which probably won't be in your path.(some of the readers might remember aimk from other platforms / applications)

If you are on a RH 8.0 box id moved libpng and zlib to .notused .. This in order to prevent version issues .. with other libpng and zlib versions.

```
export PATH=$PATH:/usr/share/pvm3/lib
export PVMROOt=/usr/share/pvm3
```

I then run aimk newunix. Then we start pvm and quit it. The daemon stays active..

And a last thing that is not known by a novice pvm user is that pvm does use its own paths, and you have to put pvmpov either in that path or launch it with the complete pathname.

```
[root@dhcp71 povray31]# /usr/local/bin/pvmpov −L
```

```
/usr/src/povray/pvmpov3_1g_2/povray31/include/ +Iskyvase.pov
+Oskyvase.tga +NT16 +NW64 +NH64 +v +w1024 +h768
Persistence of Vision(tm) Ray Tracer Version 3.1g.Linux.gcc
  This is an unofficial version compiled by:
  Jakob Flierl  − PVMPOV Version 3.1g.2
  The POV−Ray Team(tm) is not responsible for supporting this version.
Copyright 1999 POV−Ray Team(tm)
Never found section  in file
/usr/src/povray/pvmpov3_1g_2/povray31/include/.
Initializing PVMPOV
  Spawning /usr/local/bin/pvmpov with 16 PVM tasks on 1 hosts...
  ...16 PVM tasks successfully spawned.
  Waiting up to 120s for first slave to start...
  Slave 0 successfully started.
Parsing Options
  Input file: skyvase.pov (compatible to version 3.1)
  Remove bounds........On  Split unions........Off
  Library paths: /usr/local/lib/povray31 /usr/local/lib/povray31/include
Output Options
  Image resolution 1024 by 768 (rows 1 to 768, columns 1 to 1024).
  Output file: skyvase.tga, 24 bpp PNG
  Graphic display.....Off
  Mosaic preview......Off
  CPU usage histogram.Off
  Continued trace.....Off  Allow interruption...On  Pause when
done.....Off
  Verbose messages.....On
Tracing Options
  Quality:  9
  Bounding boxes.......On  Bounding threshold: 25
  Light Buffer.........On  Vista Buffer.........On
  Antialiasing........Off
  Radiosity...........Off
Animation Options
  Clock value....   0.000  (Animation off)
PVM Options
  Block Width....      64  Block Height...      64
  PVM Tasks......      16
  PVM Nice.......       5
  PVM Arch.......
  PVM Slave...... /usr/local/bin/pvmpov
  PVM WorkingDir. /usr/src/povray/pvmpov3_1g_2/povray31
Redirecting Options
  All Streams to console..........On
  Debug Stream to console.........On
  Fatal Stream to console.........On
  Render Stream to console........On
  Statistics Stream to console....On
  Warning Stream to console.......On

Starting frame 0...
 Slave 1 at dhcp71.office.be.stone−it.com successfully started.
 Slave 2 at dhcp71.office.be.stone−it.com successfully started.
 Slave 3 at dhcp71.office.be.stone−it.com successfully started.
 Slave 4 at dhcp71.office.be.stone−it.com successfully started.
 Slave 5 at dhcp71.office.be.stone−it.com successfully started.
 Slave 6 at dhcp71.office.be.stone−it.com successfully started.
 Slave 7 at dhcp71.office.be.stone−it.com successfully started.
 Slave 8 at dhcp71.office.be.stone−it.com successfully started.
 Slave 9 at dhcp71.office.be.stone−it.com successfully started.
 Slave 10 at dhcp71.office.be.stone−it.com successfully started.
 Slave 11 at dhcp71.office.be.stone−it.com successfully started.
 Slave 12 at dhcp71.office.be.stone−it.com successfully started.
 Slave 13 at dhcp71.office.be.stone−it.com successfully started.
 Slave 14 at dhcp71.office.be.stone−it.com successfully started.
 Slave 15 at dhcp71.office.be.stone−it.com successfully started.
  0:00:53 86.46 of blocks complete.Not using dhcp71.office.be.stone−it.com
for reassignment (77%)
  0:00:53 86.98 of blocks complete.Not using dhcp71.office.be.stone−it.com
for reassignment (67%)
Not using dhcp71.office.be.stone−it.com for reassignment (86%)
Not using dhcp71.office.be.stone−it.com for reassignment (85%)
  0:00:55 89.06 of blocks complete.   640 of  768 lines finished (in frame
0).Not using dhcp71.office.be.stone−it.com for reassignment (65%)
  0:00:56 91.67 of blocks complete.Not using dhcp71.office.be.stone−it.com
for reassignment (72%)
  0:00:56 92.71 of blocks complete.Not using dhcp71.office.be.stone−it.com
for reassignment (80%)
  0:00:57 93.75 of blocks complete.
Slave at dhcp71.office.be.stone−it.com has exited.
  0:00:57 94.79 of blocks complete.
Slave at dhcp71.office.be.stone−it.com has exited.

Slave at dhcp71.office.be.stone−it.com has exited.
  0:00:58 95.83 of blocks complete.
Slave at dhcp71.office.be.stone−it.com has exited.
  0:00:58 96.35 of blocks complete.   672 of  768 lines finished (in frame
0).Not using dhcp71.office.be.stone−it.com for reassignment (77%)

Slave at dhcp71.office.be.stone−it.com has exited.
  0:00:58 97.14 of blocks complete.   688 of  768 lines finished (in frame
0).
Slave at dhcp71.office.be.stone−it.com has exited.
  0:00:59 97.92 of blocks complete.
Slave at dhcp71.office.be.stone−it.com has exited.
  0:00:60 98.44 of blocks complete.   704 of  768 lines finished (in frame
0).
Slave at dhcp71.office.be.stone−it.com has exited.
  0:01:03 100.00 of blocks complete.   768 of  768 lines finished (in
frame 0).
Finishing frame 0...rtw. 768


Waiting for remaining slave stats.


PVM Task Distribution Statistics:
           host name  [ done ] [ late ]           host name  [ done ] [
late ]
dhcp71.office.be.stone−it.com  [ 5.21%] [
0.00%]dhcp71.office.be.stone−it.com  [ 7.81%] [ 0.07%]
dhcp71.office.be.stone−it.com  [ 8.85%] [
1.17%]dhcp71.office.be.stone−it.com  [ 4.69%] [ 0.00%]
dhcp71.office.be.stone−it.com  [ 8.85%] [
0.98%]dhcp71.office.be.stone−it.com  [ 4.17%] [ 0.00%]
dhcp71.office.be.stone−it.com  [ 5.21%] [
0.00%]dhcp71.office.be.stone−it.com  [ 8.33%] [ 0.52%]
dhcp71.office.be.stone−it.com  [ 5.21%] [
0.00%]dhcp71.office.be.stone−it.com  [ 5.73%] [ 0.72%]
dhcp71.office.be.stone−it.com  [ 7.29%] [
2.73%]dhcp71.office.be.stone−it.com  [ 4.17%] [ 0.00%]
dhcp71.office.be.stone−it.com  [ 5.21%] [
0.00%]dhcp71.office.be.stone−it.com  [ 6.77%] [ 0.13%]
dhcp71.office.be.stone−it.com  [ 4.69%] [
0.00%]dhcp71.office.be.stone−it.com  [ 7.81%] [ 0.00%]


POV−Ray statistics for finished frames:
skyvase.pov Statistics (Partial Image Rendered), Resolution 1024 x 768
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Pixels:          303104   Samples:          303104   Smpls/Pxl: 1.00
Rays:           1192710   Saved:                 0   Max Level: 0/5
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Ray−>Shape Intersection          Tests       Succeeded  Percentage
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Cone/Cylinder                  1842227          900504     48.88
CSG Intersection               2742731          323346     11.79
CSG Union                      1801008          521692     28.97
Plane                         20223278        11233348     55.55
Quadric                        1801008         1196533     66.44
Sphere                         1801008          461786     25.64
Bounding Object                1842227          900504     48.88
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Calls to Noise:            1201944   Calls to DNoise:        2108954
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Shadow Ray Tests:          2856188   Succeeded:                85620
Reflected Rays:             889606
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Smallest Alloc:                  9 bytes   Largest:            20508
Peak memory used:          5643343 bytes
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Time For Trace:    0 hours  1 minutes   7.0 seconds (67 seconds)
    Total Time:    0 hours  1 minutes   7.0 seconds (67 seconds)
```

As you can see the application was splitted into differen parts and run separatly, openMosix then did the job of balancing the load to other machines.

I had good results with 2 to 3 times the the number of cpu's I had available

Chapter 17. BioInformatics and openMosix

17.1. Introduction

17.2. Blast

One of the more frequent used application in this field is Blast, *Blast has a patch available that makes it work smoother* with openMosix, but that's not the only alternative.

First of all there are some known problems with this patch, and other versions of blast , blast tends to segfault sometimes, this mostly happens with the preformatted databases you download from the internet. If you run formatdb on a raw database these errors tend to go away.

Next to the openMosix blast patch a lot of people run *MPIBlast* Given the fact that openMosix tends to speed up MPI, adding openMosix to this config might even give you more power for your money, however we will have to do some extra research to be able to confirm this.

V. openMosix Development

Table of Contents

Chapter 18. Getting started with openMosix internals

18.1. Introduction

this part has been written by Amit Shah

There's not much documentation available right now for the kernel. I hope to write some in the coming weeks. Anyways, here's how the sources are laid out:

The openMosix code resides largely in hpc/ and include/hpc. There are lots of patches to the core kernel files everywhere, right from the arch/i386 directories to mm/, fs/, etc. You need to read up the code which interests you and think that would matter for the present situation (that shouldn't be a problem, since you've done kernel coding).

here's what you should expect in each of the source files:

- hpc/badops.c: one file to handle all the bad operations: mostly return err codes
- hpc/balance.c: The load balancer code (load + mem usage + n/w usage)
- hpc/comm.c: The intra−cluster communication setup
- hpc/config.c: The config code for openMosix: after you run the startup script
- hpc/decay.c: decay (age) the stats and info collected from other nodes
- hpc/deputy.c: Code executed on the deputy: service remote syscalls (ie. after the process has migrated), signals, etc.
- hpc/dfsa.c: Direct File System Access code: the distributed file system abstraction layer
- hpc/div.c: the algorithms to do floating point divisions
- hpc/export.c: export symbols needed in other files
- hpc/freemem.c: to keep track of free, avl. memory and to free it if need be. hugely taken from the Linux mm/ code.
- hpc/hpcadmin.c: tune openMosix admin values (through /proc/hpc)
- hpc/hpcproc.c: The /proc/hpc code is handled here
- hpc/info.c: The info daemon: sends and receives (multicast) load+mem usage stats throughout the cluster
- hpc/init.c: Initialization code: initializes the daemons, etc.
- hpc/kernel.c: most of the "core" code: all the important algorithms, decisions, etc. made here.
- hpc/load.c: calculation of local load, etc.
- hpc/mig.c: Code that handles the migration. Code in this file is invoked on any migration: deputy−>remote, remote−>deputy; remote−>remote
- hpc/prequest.c: handles the process's requests: signals, more memory, etc.
- hpc/remote.c: Code executed when the process is on the remote: syscalls handling on remote, passing control to deputy, etc.
- hpc/rinode.c: fs/ related stuff: used mostly for DFSA
- hpc/service.c: setting up daemons, getting memory, etc.
- hpc/syscalls.c: handles all the remote syscalls here
- hpc/ucache.c: handles the ucache: mostly mm/, fs/ stuff.

the other files like auto_syscalls.c, alternate.c are generated at compile time.

VI. FAQ

Table of Contents

Chapter 19. the openMosix FAQ

19.1. General

19.1.1. What is openMosix?:

The openMosix system is a Linux kernel extension for single−image clustering. It extends the outstanding MOSIX project, but is instead licensed under the GNU General Public License (GPL).

19.1.2. What does the term single−image clustering mean?:

There are many varieties of clusters, and a single−image cluster has multiple copies of a single operating system kernel.

19.1.3. Does openMosix have a homepage?:

Yes. It is at www.openMosix.org. The SourceForge project page is at *http://openmosix.sf.net/*

19.1.4. Are there any mailing lists for openMosix?:

Yes. There are three:

1. For general discussion, use mailto:openmosix−general@lists.sourceforge.net, whose general information page is at *http://lists.sourceforge.net/lists/listinfo/openmosix−general*
2. For developers, use mailto:openmosix−devel@lists.sourceforge.net, whose general information page is at *http://lists.sourceforge.net/lists/listinfo/openmosix−devel*
3. Italian Language openMosix Mailing List hosted by Democritos (the INFM National Simulation Center in Trieste)

19.1.5. Can I contribute to the openMosix project?:

Yes. The openMosix effort already has more than 10 contributors. Unlike the Linux kernel maintenance system, Moshe Bar appoints official maintainers and then gives these maintainers the commit bit to the openMosix CVS source tree, similarly to FreeBSD.

Right now we are looking for more experienced kernel hackers to work on new features like checkpoint/restart.

Write to mailtomoshe@moelabs.com if you would like to become an openMosix developer.

19.1.6. Who is the copyright holder of openMosix?:

All MOSIX code is copyright by Professor Amnon Barak of Hebrew University of Jerusalem. All openMosix code is copyright by Moshe Bar, Tel Aviv. The openMosix system does not contain any non−GPL (i.e. MOSIX) code.

19.1.7. Is openMosix a fork of MOSIX?:

Originally, openMosix was a fork of MOSIX, but it has evolved into an advanced clustering platform. The openMosix system no longer contains any non−GPL (i.e. MOSIX) code.

Compared to MOSIX, a number of features were added:

A port to the UML (User−mode Linux) architecture
New and cleaner migration code
A better load balancer
Much reduced kernel latencies
Support for Dolphin and IA64
A greatly simplified installation processes that uses RPM packaging
A wealth of documentation

19.1.8. Why did openMosix split from the MOSIX group?:

The principal issue was that MOSIX was not licensed with an Open Source license.

19.2. Getting, building, installing and running openMosix

19.2.1. Where do I get openMosix?:

The RPMs and source for openMosix are available from our Downloads/Files Section. Please read the release notes first!

Also, Gentoo Linux's emerge sys−apps/openmosix−user and Debian GNU/Linux openMosix packages are available.

19.2.2. Can I mix MOSIX and openMosix nodes in the same cluster?:

No. Just like the older MOSIX, you should not mix nodes because the protocols are subject to unannounced changes from version to version. In addition, every new version has bug fixes which warrant updating to the new kernels.

19.2.3. How do I build openMosix?:

1. Start by unpacking both the Linux kernel sources and the corresponding openMosix distribution in a directory, say /usr/src.
2. Then

   ```
   $ cd /usr/src; tar xzf linux−2.x.xx.tar.gz ;gunzip openMosix2.x.xx.gz
   ```
3. Apply the openMosix patches to the pristine Linux kernel sources with

   ```
         $ patch −p1 openMosix2.x.xx−x
   ```

   The directory /usr/src/linux−2.x.xx now contains the 2.x.xx kernel sources with the openMosix patches. Compile and install the resulting kernel as usual.

19.2.4. What are userland tools?:

Userland tools are a collection of administrative tools used to examine and control an openMosix node.

19.3. Kernel Questions

19.3.1. What kernel versions does openMosix support?:
19.3.2. I'm trying to compile an openMosix−patched kernel. What compiler version should I use?:
19.3.3. I've compiled the kernel from the sources. How do I add it to the bootloader (LILO, GRUB, other)?:
19.3.4. I installed a Linux distribution and it says that its kernel is x.x.x−x. The openMosix README says not to mix kernel versions. Does that mean that the openmosix−x.x.x−y RPM will not work on my machine?:
19.3.5. What does the phrase the same kernel on every machine mean? Does it mean the same kernel version, or the same kernel image?:

19.3.1. What kernel versions does openMosix support?:

The latest Linux kernel supported is 2.4.19. Later versions of the 2.4 series will be supported, as will kernel versions in the 2.5 series.

19.3.2. I'm trying to compile an openMosix−patched kernel. What compiler version should I use?:

You should use gcc−2.95.3 as this is the recommended compiler for 2.4 kernels. This is a Linux kernel requirement, not just an openMosix requirement. However, nothing precludes you from having, on the same system, gcc−2.95.3 for kernel compiles and gcc−3.x for non−kernel compiles.

Additional notes: There are many kernel−related issues with gcc−3.x compilers. Inlining, optimization and page alignment do strange things to operating systems kernels. The standard Linux kernel is only guaranteed to compile and work properly with gcc 2.95.3.

However, the Red Hat gcc 2.96 compiler is 2.95 + RH patches. In this case, you should ensure you use gcc−2.96−74 or later. gcc−2.96−54 will not build the kernel correctly. In addition, please pay attention to compiler optimization. Anything greater than −O2 may not be wise. Similarly, if you choose to use gcc−2.95.x or derivatives, be sure not to use −fstrict−aliasing (which, depending on your version of gcc 2.95.x, may necessitate using −fno−strict−aliasing).

19.3.3. I've compiled the kernel from the sources. How do I add it to the bootloader (LILO, GRUB, other)?:

Treat an openMosix kernel just like any other kernel. The openMosix system is simply an extension to the kernel, and will be treated like a standard kernel by your bootloader.

19.3.4. I installed a Linux distribution and it says that its kernel is x.x.x−x. The openMosix README says not to mix kernel versions. Does that mean that the openmosix−x.x.x−y RPM will not work on my machine?:

No. It means is that if you install openMosix on your cluster, all your machines should have the openmosix−x.x.x−y kernel installed. You should not mix kernels which have different kernel versions, i.e. do not mix openmosix−x.x.z−x, and openmosix−x.x.x−y, etc.

19.3.5. What does the phrase the same kernel on every machine mean? Does it mean the same kernel version, or the same kernel image?:

It means the same kernel version. You can build different kernel images of the same source version to meet the hardware/software needs of a given node.

19.4. File Systems

19.4.1. What's oMFS, how do I use, and where do I get it?:

The openMosix File System (oMFS) is the filesystem used by openMosix kernels. You get it by installing an openMosix kernel on the nodes of your cluster with oMFS enabled in the kernel−config. (It should be enabled in the openMosix RPMs by default.)

You should also enable Direct Filesystem Access (DFSA) which allows a migrated process to execute many syscalls on the remote node locally without the need to migrate it back to its home node.

The use and administration of oMFS is very similar to NFS, but unlike NFS, oMFS features:

- Cache consistency
- Timestamp consistency
- Link consistency

The DFSA layer on top of oMFS makes sure to move the process to the data, instead of vice versa, whenever it makes sense.

Please read more about oMFS and how to use it in earlier chapters of the HOWTO.

19.4.2. Can somebody explain to me the difference bewteen MFS and DFSA, and why I would need DFSA?:

DFSA stands for Direct File System Access and is an optimization. It allows remote proccesses to perform some file system system calls locally rather then sending them to their home node. MFS stands for Mosix File System and allows all nodes access to all node filesystems. DFSA runs on top of a cluster filesystem, in this case MFS.

19.5. Programming openMosix

19.5.1. Generally, how do I write an openMosix−aware program?:

Write your programs as you normally would. Any processes that you spawn are candidates for migration to another node.

19.5.2. Can I write openMosix programs in perl?:

Yes. Use the Parallel::ForkManager available from CPAN or directly from http://www.cpan.org/authors/id/D/DL/DLUX/Parallel−ForkManager−x.x.x.tar.gz.

19.6. Resources

19.6.1. Where can I find out technical details about openMosix?:

Here are some links:

- Brian Pontz's openMosix source code browser is at http://www.openmosix.org/lxr/source.
- openMosix Internals: How openMosix Works is at http://sourceforge.net/docman/display_doc.php?docid=10390&group_id=46729.
- LTSP + openMosix: A Quick How−To is at http://sourceforge.net/docman/display_doc.php?docid=10431&group_id=46729.
- Distributed OSs: General description of openMosix is at http://sourceforge.net/docman/display_doc.php?docid=9562&group_id=46729.

More links at the link section of the howto.

19.6.2. What other resources are available?:

Here are some:

- The goal of the Parallel Execution Framework is to create a standalone device that allows easy clustering of standard PCs. See http://parallel.sourceforge.net.

19.7. Miscellaneous

19.7.1. I don't see all my nodes. What's happening?:

When you run 'mosmon', press 't' to see the total number of machines running. Does it warn you that openMosix is not running?

If it does, then make sure your machine's IP address is included in /etc/openmosix.map. Don't use 127.0.0.1. If you do, you will probably have problems with your DHCP server or your DNS nameserver.

If it does not, then see what machines show up. Do you see only your machine?

If yes, then your machine is most likely running a firewall and is not letting openMosix through.

If not, then the problem is most likely with the machine that doesn't show up.

Also: Do you have two NIC cards on a node? If so, you have to edit /etc/hosts to have a line that has the following format

```
<non−cluster ip> <cluster−hostname>.<cluster−domain> <cluster−hostname>
```

You might also need to set up a routing table, which is an entirely different subject.

Maybe you used different kernel−parameters on each machine? Especially if you use the 'Support clusters with a complex network topology' option you should take care that you use the same value for the also appearing option 'Maximum network−topology complexity support' on each machine.

19.7.2. Whats the difference between /etc/mosix.map , /etc/hpc.map , /etc/openmosix.map:

They represent three stages of Mosix/openmosix growth. The file /etc/mosix.map is the orginal Mosix map name, The file /etc/hpc.map was an early openMosix map name (and 'hpc' is still used for the /proc files in openMosix). The current map name is /etc/openmosix.map.

19.7.3. setpe: the supplied table is well−formatted, but my IP address (127.0.0.1) is not there!:

You'll need to modify your /etc/hosts file. On Red Hat machines mostly the /etc/hosts file includes a line like

```
127.0.0.1 hostname.domain.com localhost
```

If hostname.domain.com has an IP address of 192.168.10.250, and if you looked up hostname.domain.com you might get 127.0.0.1 as an answer.

However, if you put

```
192.168.10.250 hostname.domain.com
127.0.0.1 localhost
```

in your /etc/hosts, openMosix won't complain.

19.7.4. I want to install openMosix but I am afraid my machines are too weak for this:

A machine is never too weak: I have three P200s (64MB each) and two P166s (one with 48MB and one with 192MB). Two of them are on 10Base−T and the other three on 100Base−T. Even with these antiquated machines and "heterogenous" network, I get perfect load balancing to run simulation programs that I write in Perl. (Look at our ProgramToTestACluster"). Don't be held back by the fact your machines are old. To us this is a nice feature of openMosix: you can add newer machines to an existing cluster as they become available.

And you do not need to have all identical machines. That's fantastic!

However, a 100Base−T network is recommended! Contributed by Charles Nadeau.

19.7.5. Under what conditions does VMWare work with openMosix:

If you intend to run VMWare under openMosix so that openMosix would load−balance several instances of that (yes, that works). But, if you want to run openMosix in several VMWare instances and let these instances load balance (that fails).

The first case works. The latter case does not work because VMware has a bug in its Pentium emulation that makes VMware crash (not openMosix, but the VMware binary) on the first migration.

19.7.6. What architectures besides x86 (e.g. SPARC, AXP, PPC...) are supported by openMosix?:

Only IA−32 is currently supported. The port of openMosix to the Intel(r) Itanium(tm) IA−64 Processor Family is complete. Project plans for openMosix' second year include porting to the 64−bit AMD Opteron(tm) processor.

19.7.7. Is there a parallel make tool for openMosix such as MPmake?:

You can use regular gcc make. just use make −j #, where the # represents how many child proccesses to spawn.

Chapter 20. PlumpOS FAQ

20.1. Frequently Asked Questions

20.1.1. What is PlumpOS?

PlumpOS is a mini linux distribution aimed at an easy way to add nodes to an openMosix cluster without a lot of work or even thought. The original idea came from (and current development is inspired by) ClumpOS, the now deceased project by Jean−David Marrow with a very similar goal for MOSIX clusters.

20.1.2. Why would I use PlumpOS?

The first reason to use PlumpOS is to try out openMosix clusters in a simple and convenient fashion. The second reason is to add nodes to an existing cluster without a lot of work. PlumpOS is in no way a replacement for enterprise clusters nor does it strive to be some kind of super duper cluster project; it merely presents a quick and easy way to turn your average 586+ PC into another node for your cluster.

20.1.3. How does it work?

From a developer's standpoint it's somewhat difficult but to the common user it's quite simple: simply burn an ISO with all the required files on it, pop it into a computer's CDROM drive, boot up the PC (making sure there's a working network card and cable installed) and there you have one more computer to distribute load/forked processes to. Because it's a child/slave node, you have to already have an openMosix box up and running to run tasks for PlumpOS computers to process. Without a master node, PlumpOS is pretty useless. That computer or some other network device must already be serving DHCP addresses or you'll find yourself configuring all your PlumpOS nodes by hand (which won't make for a happy network administrator − but then again that's why we have junior admins ^_^). Also, make sure you have the openMosix autodiscovery daemon (aka omdiscd) running on your master node. The autodiscovery daemon can be found into openMosix's userspace−tools package.

20.1.4. Where can I go for more help?

Well, for PlumpOS specific help you should join the PlumpOS mailing list at *http://lists.sourceforge.net/lists/listinfo/plumpos−general* and ask a question (that wasn't answered on this FAQ) or go to *irc.freenode.net #PlumpOS*. For openMosix related questions you should join the openMosix−general or openMosix−devel mailing list at *http://openmosix.sourceforge.net/mailing.html* or visit *irc.freenode.net #openMosix*.

20.1.5. Where is your homepage?

The official PlumpOS homepage is at *http://PlumpOS.sourceforge.net/*. You will find updated versions of this document there, as well as updated news, info, etc.

20.1.6. The penguins are gonna eat the poor halibut in the logo!

Yes, but not to worry: no halibuts were harmed in the making of the logo. The halibut was found deceased on an ice drift after having suffered a heart attack. Also, any resemblance the halibut may have to another operating system's mascot of choice is purely coincidental.

Appendix A. How to produce openMosix's Kernel RPM files

A.1. How to produce openMosix's Kernel RPM files

A step−by−step guide for the dumb release manager and the adventurous "do it yourself" RPM packager , by Mirko Caserta

1. Install a RedHat 8 (psyche) on your machine. This is so far the platform used to produce the rpms and it's known to do its job well
2. Get an updated copy of the "linux−openmosix" module from oM's CVS repository − details can be found at http://sourceforge.net/cvs/?group_id=46729
3. Get the tarball for the Linux Kernel sources which we are going to patch and put it in /usr/src/redhat/SOURCES − supposing we're talking about a 2.4.20 Kernel, get the linux−2.4.20.tar.bz2 file from one of the many http://www.kernel.org/ mirrors worldwide
4. Unpack the kernel tarball under /usr/src, ie:

   ```
   # cd /usr/src
   # tar vxjf redhat/SOURCES/linux−2.4.20.tar.bz2
   ```
5. Make a symbolic link to the directory where you've checked out the linux−openmosix module, for instance:

   ```
   # ln −s /home/mcaserta/src/linux−openmosix/linux−openmosix \
                /usr/src/linux−openmosix
   ```
6. copy the .spec file and all the .config files which are found in this directory under /usr/src/redhat/SOURCES, ie:

   ```
   # cp /usr/src/linux−openmosix/configs/openmosix−kernel.spec \
        /usr/src/redhat/SOURCES

   # cp /usr/src/linux−openmosix/configs/*.config \
        /usr/src/redhat/SOURCES
   ```
7. Now it's time to check the version numbers before we make the patch file: make sure the very first lines in

   ```
   /usr/src/linux−openmosix/Makefile and
   /usr/src/redhat/SOURCES/openmosix−kernel.spec have the correct
   ```

   kernel version and openMosix revision number
8. Good, time to make the patch (suppose we're releasing a patch for the 2.4.20 Linux Kernel and the 3rd release of openMosix):

   ```
   # cd /usr/src
   # diff −Naur −−exclude=CVS −−exclude=configs \
          linux−2.4.20 linux−openmosix > \
          /usr/src/redhat/SOURCES/openMosix−2.4.20−3
   # bzip2 /usr/src/redhat/SOURCES/openMosix−2.4.20−3
   ```
9. At this point your /usr/src/redhat/SOURCES directory should look like:

   ```
   # ls /usr/src/redhat/SOURCES
   kernel−2.4.20−athlon.config      kernel−2.4.20−i686−smp.config
   kernel−2.4.20−athlon−smp.config  linux−2.4.20.tar.bz2
   kernel−2.4.20−i386.config        openMosix−2.4.20−3.bz2
   kernel−2.4.20−i686.config        openmosix−kernel.spec
   ```
10. Now you only need to rpmbuild the whole thing − what I usually do is:

    ```
    # cd /usr/src/redhat/SOURCES
    # rpmbuild −ba −−target   i386   openmosix−kernel.spec
    # rpmbuild −bb −−target   i686   openmosix−kernel.spec
    # rpmbuild −bb −−target athlon   openmosix−kernel.spec
    ```

but you can easily build all rpms by calling:

```
 # rpmbuild −ba −−target all_x86  openmosix−kernel.spec
```

11. After rpmbuild has done its job, you should have the following files under the /usr/src/redhat directory:

```
  a) RPMS/i386/openmosix−kernel−2.4.20−openmosix3.i386.rpm
    b) RPMS/i686/openmosix−kernel−2.4.20−openmosix3.i686.rpm
    c) RPMS/i686/openmosix−kernel−smp−2.4.20−openmosix3.i686.rpm
    d) RPMS/athlon/openmosix−kernel−2.4.20−openmosix3.athlon.rpm
    e) RPMS/athlon/openmosix−kernel−smp−2.4.20−openmosix3.athlon.rpm
    f) RPMS/i386/openmosix−kernel−source−2.4.20−openmosix3.i386.rpm
    g) SRPMS/openmosix−kernel−2.4.20−openmosix3.src.rpm
    h) SOURCES/openMosix−2.4.20−3.gz
```

where:

```
  a) binary kernel package for i386 UP* machines
    b) binary kernel package for i686 UP* machines
    c) binary kernel package for i686 SMP** machines
    d) binary kernel package for athlon UP* machines
    e) binary kernel package for athlon SMP** machines
    f) source kernel package for any x86 machine (basically this is
       useful if you need to have the openMosix kernel headers)
    g) source kernel package (see point 11)
    h) kernel patch file compressed with gzip
```

12. The magic spell to obtain all the files back from the .src.rpm file is: # rpm2cpio openmosix−kernel−....src.rpm | cpio −di

Special thanks to Martin Høy for the help while I was trying to get the whole thing together.

I hope you find this document useful. At least it is for me since I tend to forget things a few minutes after I've accomplished them :)

* UP = UniProcessor (i.e. one CPU) ** SMP = Symmetric Multi Processing (i.e. more than one CPU)

Appendix B. More Info

B.1. irc

Some of the openMosix enthousiasts spend time online helping out people on irc. We are on irc.freenode.net on #openMosix. Just join us there to dicuss your problems , ideas and other stuff about openMosix

B.2. Further Reading

B.3. Translations

Some people have been working on partial translations of this HOWTO, or just plain openMosix documentation in their own language.

If you are working on a translation of this document let us know.

B.3.1. Chinese

Ding Wei has written some documents in *Chinese*, you can read them at *http://software.ccidnet.com/pub/disp/Article?columnID=732&articleID=25795&pageNO=1*

Here is a *local* copy to the Chinese doc in PDF

B.3.2. Spanish

Together with some collegues Miquel Catalán Coïthas been working on a spanish translation of the HOWTO *http://w3.akamc2.net/*

B.3.3. Russian

Dmitry Katusubo translated the openmosix website and together with Yuri Prushinsky he also translated the openMosix HOWTO *http://www.openmosix.org.ru/docs/openMosix−HOWTO−multi/*

B.4. Links

- *openMosix on Sourcefourge*
- *the openMosix HOWTO*
- *the openMosix FAQ*
- *the openMosix WIKI*
- *the openMosix Community Contributions*
- *openmosixview*
- *Mosix Debian HOWTO*
- *K12 LTSP Mosix HOWTO*
- *Mosix Mandrake Linux Terminal Server Project*
- *openMOSIXVIEW, a GUI for managing openMosix−Cluster*
- *User Mode openMosix, a virtual openMosix cluster running in User−mode*

- *RxLinux, Web Interface for central configuration and management*
- *LTSP+OpenMosix: A Mini How−To*
- *FuBAR: An openMosix cluster at Texas AM − Corpus Christi*
- *Charting the Land of Elliptic Curves using openMosix*
- *openMosixWebView*
- *Italian experiences with openMosix clusters* Links to several papers presented at the conference. These papers cover many recurring questions seen on the openMosix mailing list:

    - Osservatorio Astronomico Cagliari (pdf)*Securing an openMosix cluster in an untrusted networking environment.*
    - D.T.I. dell'Università di Milano, Polo didattico e ricerca, Crema (pdf) *Experiences with Java and C programs on openMosix.*
    - Riello Group (pdf) *Use of openMosix for parallel I/O balancing on storage (backup)*
    - INFM & Dept. of Physics University of Napoli (pdf) *Documents a number of small to large clusters and their uses.* (Includes many references to software and projects.)
    - Conecta srl (pdf) *Contains more information on Kraken, the openMosix based game server. Included is how they monitor the cluster's temperature via lmsenors and how they improved internode communications using iproute2 queue controls.*
- *openMosix with Diskless client*
- *Use networked Linux systems to solve your computing challenges by Daniel Robbins*

B.5. Mailing List

- *openMosix mailing list*
- *openmosix−view mailing list*
- *ClumpOS mailing list*

Appendix C. Credits

The list of people who deserve credits for this HOWTO is long, I actually lost track of all the people that should be in here. I often add their names right next to the parts they have contributed.

If you feel that your name is missing here do not hesitate to contact me and I'll gladly put your name into the list.

Scot W. Stevenson

I have to thank Scot W. Stevenson for all the work he did on this HOWTO before I took over. He made a great start for this document.

Assaf Spanier

worked together with Scott in drafting the layout and the chapters of this HOWTO. and now promised to help me out with this document.

Matthias Rechenburg

Matthias Rechenburg should be thanked for the work he did on openMosixview and the accompanying documentation , which we included in this HOWTO.

Jean−David Marrow

is the author of Clump/OS, he contributed the documentation on his distribution to the HOWTO.

Bruce Knox

is the maintainer of the openMosix website, he helps where he can and gives a lot of feedback !

Evan Hisey

for putting a lot of effort into putting extra documentation in the WIKI

Charles Nadeau

for putting a lot of effort into putting extra documentation in the WIKI

Louis Zechter

Moshe Bar

For writing the code he wrote and helping out with the docs wherever he knows the answers !

Amit Shah

for getting started with the openMosix internals

Mirko Caserta

For sending in huge patches to this howto

Ramon Pons

for proofreading the howto and sending in some advice

Appendix D. GNU Free Documentation License

Version 1.1, March 2000

> Copyright (C) 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111−1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written 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 non−commercially. 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 that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

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. (For example, 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.

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 "Transparent" copy of the Document means a machine−readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and 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 has been designed to thwart or discourage subsequent modification by readers is not Transparent. 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 designed for human modification. Opaque formats include PostScript, PDF, 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 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.

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 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 publicly−accessible computer−network location containing a complete Transparent copy of the Document, free of added material, which the general network−using public has access to download anonymously at no charge using public−standard network protocols. 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 less than five).
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", and 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. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, 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 as "Endorsements" or to conflict in title with any Invariant Section.

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.

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, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self−contained works thus compiled with the Document, on account of their being thus compiled, if they 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 quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around 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 holders, 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 provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

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.

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:

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front−Cover Texts being LIST, and with the Back−Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front−Cover Texts, write "no Front−Cover Texts" instead of "Front−Cover Texts being LIST"; likewise for Back−Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index