(Open)Mosix Experience in Naples

Rosario Esposito1, Francesco M. Taurino1,2, Gennaro Tortone1
1 (Istituto Nazionale di Fisica Nucleare, via Cintia – 80126 Napoli, Italy)
2 (Istituto Nazionale di Fisica Nucleare – UdR di Napoli, via Cintia – 80126 Napoli, Italy)

This document is a short report of the activities carried out by the computing centre of INFN, INFM and Dept. of Physics at University of Naples "Federico II", concerning the installation, management and usage of HPC Linux clusters running (open)Mosix [1][2].
These activities started on small clusters, running a local copy of the operating system and evolved through the years to a configuration of computing farms based on a diskless architecture, simplifying installation and management tasks.

Introduction

Thanks to high performances offered by low cost systems (common personal computers) and availability of operating systems like Linux, we were able to implement HPC clusters using standard PCs, connected to an efficient LAN.
This solution offers several advantages:

- lower costs than monolithic systems;
- common technologies and components available trough the retailers (COTS, Components Off The Shelf);
- high scalability.

Mosix in Naples

Our first test cluster was configured in January 2000.
At that time we had 10 PCs running Linux Mandrake 6.1, acting as public Xterminals.
They were used by our students to open Xsessions on a DEC-Unix AlphaServer to compile and execute simple simulation programs.
Those machines had the following hardware configuration:
- Pentium 200 Mhz
- 64 MB RAM
- 4 GB hard disk

Each computer, with the operating system installed on the local disk, was connected to a 100 Mb/s Fast Ethernet switch.
We tried to turn this "Xterminals" in something more useful. Mosix 0.97.3 and kernel 2.2.14 were used to convert those PCs in a small cluster to perform some data-reduction tests (mp3 compression with *bladeenc*[1] program).
Compressing a wav file in mp3 format using *bladeenc* could take up to 10 minutes on a Pentium

[1] *http://bladeenc.mp3.no/*

200. Using the Mosix cluster, without any source code modification, we were able to compress a ripped audio cd (14-16 songs) in no more than 20 minutes.

Once verified the ability of Mosix to reduce the execution time of those "toy" programs thanks to preemptive process migration and dynamic load balancing, we decided to implement a bigger cluster to offer a high performance facility to our scientific community.
The first step was to allow our students to directly log on the cluster to submit their jobs. In this way the machines started to be used as computing nodes rather than simple Xterminals.

Majorana

The students' Linux farm was a success, then we decided to build a more powerful Mosix cluster, named "Majorana", available to all of our users, in line with the nowadays trends in operating systems and development platforms (Linux + X86), using low cost solutions and opensource tools (MOSIX, MPI, PVM).

The farm was composed by 6 machines.
We bought the following hardware:
5 computing elements with:

- Abit VP6 motherboard
- 2 Pentium III @800 Mhz
- 512 MB RAM PC133
- a 100 Mb/s network card (3com 3C905C)

1 server with:

- Asus CUR-DLS motherboard
- 2 Pentium III @800 Mhz
- 512 MB RAM PC133
- 4 IDE HD (os + home directories in RAID)
- a 100 Mb/s network card (3com 3C905C) - public LAN
- a 1000 Mb/s network card (Netgear GA620T) - private LAN

All of the nodes were connected to a Netgear switch equipped with 8 Fast Ethernet ports and a Gigabit port dedicated to the server.

Configuring a farm presents some difficulties due mainly to the high number of hardware elements to manage and control.
Farm setup can be a time-consuming and difficult task and the probability to make mistakes during the configuration process gets higher when the system manager has to install a large number of nodes.

Moreover, installing a new software package, upgrading a library, or even removing a program, become heavy tasks when they have to be replicated on every computing node.
A farm, at last, can be a complex "instrument" to use, since executing parallel jobs often requires an "a priori" knowledge about hosts status and a manual resources allocation.

The software architecture of the clusters we have implemented tries to solve such problems, offering an easier way to install, manage and share resources.
The model we have chosen is based on diskless nodes, using no operating system disk. One of the machines (master node) exports to all the other nodes the operating system and the applications.

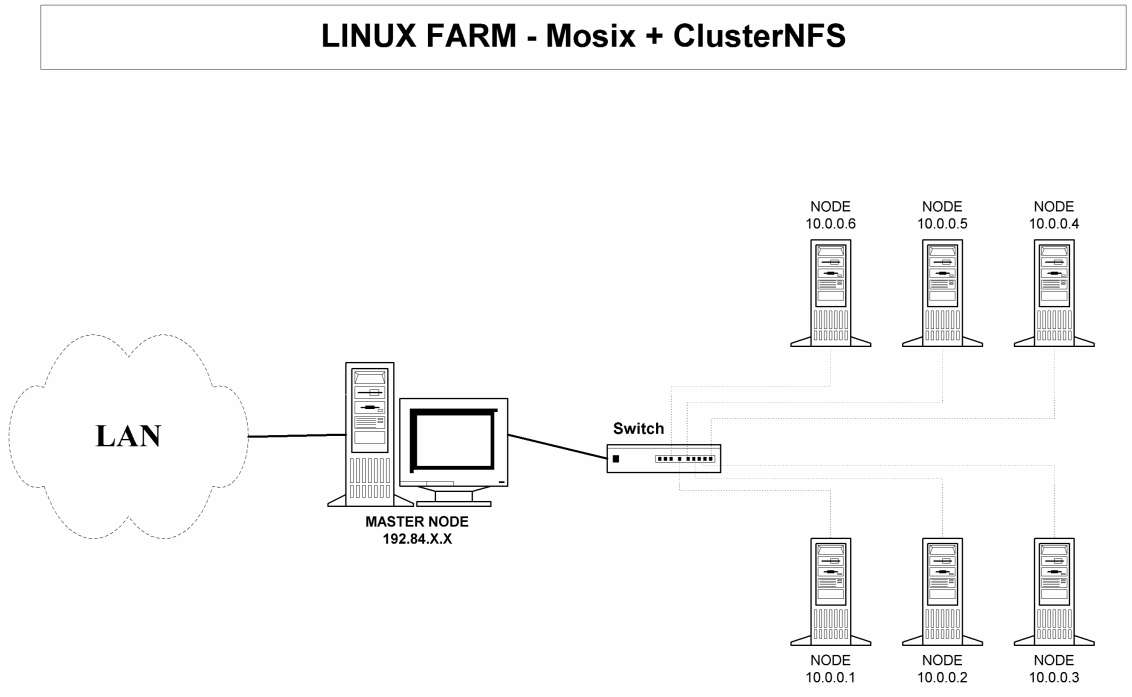

Figure 1: ***Majorana* farm configuration**

A private network is used by slave nodes to remotely execute the bootstrap procedure and to exchange data with the master node (NFS, process migration, MPI).
Each machine, except the master node, has no system disk. This solution offers several advantages:

- installing and upgrading software on the farm gets very simple as every modification involves only the master node;
- increased reliability, because there are less mechanical components;
- reduced cost, because only the master node mounts a local hard disk;

This type of setup significantly reduces the management time for "slave" nodes.

Network boot of diskless nodes is achieved using EtherBoot[2] [3], an opensource tool that allows to create a ROM containing a startup code, depending on the Ethernet card installed on the diskless machines. This code can be loaded on a boot device, such as a floppy, a hard disk, or a network adapter EERPOM.

We can shortly explain the boot sequence of a generic diskless node:

1. startup code, generated by EtherBoot, is loaded and executed at boot time
2. the node sends on the network (via bootp or dhcp) a request to obtain an IP address from the server
3. once obtained the IP address, the node downloads its MOSIX kernel from the server, using TFTP protocol
4. the MOSIX kernel is loaded and the node begins the real operating system boot
5. kernel, previously compiled with the *"root filesystem over NFS"* option, mounts its root filesystem via NFS from the server
6. once mounted the root filesystem, the node completes its boot sequence initialising system services and, eventually, mounting local disks

[2] *http://www.etherboot.org*

Since every diskless node has to mount its root filesystem via NFS, master node should export an independent root filesystem image to each client.
Exporting the same root filesystem to every client is not convenient for obvious reasons, for example configurations and log files have to be different for each machine.

To solve this problem we chose to install ClusterNFS[3] [4], an enhanced version of standard NFS server.
ClusterNFS is a an opensource tool which allows every machine in the cluster, (included the master node), to share the same root filesystem, paying attention to some syntax rules:

- all files are shared by default;
- an "xxx" file common to every client, but not to the server, has to be named "xxx$$CLIENT$$";
- an "xxx" file, for a specific client has to be named "xxx$$HOST=hostname$$" or "xxx$$IP=111.222.333.444.", where "hostname" and "111.222.333.444" are the real hostname or the ip address of that node.

This diskless cluster, currently running OpenMOSIX 2.4.19, is used by our users (mainly students and researchers) as a test platform to develop scientific applications and to submit cpu-intensive jobs, such as Monte Carlo simulations.

VIRGO

VIRGO [5] is the collaboration between Italian and French research teams, for the realization of an interferometric gravitational wave detector.
The Virgo project consists mainly in a Michelson laser interferometer made of two orthogonal arms being each 3 kilometres long. Multiple reflections between mirrors located at the extremities of each arm extend the effective optical length of each arm up to 120 kilometres. Virgo is located at Cascina, near Pisa on the Arno plain.
The main goal of the VIRGO project is the first direct detection of gravitational waves emitted by astrophysical sources.

Interferoretric gravitational wave detectors produce a large amount of "raw" data that require a significant computing power to be analysed.
Moreover a large number of computing resources has to be allocated to analyse simulated data; this is a very important task to develop new efficient analysis procedures that will be used on the real data.

To satisfy such a strong requirement of computing power we decided to build a Linux cluster running MOSIX.
The prototype of this Linux-farm [6][7][8] is located in Naples and consits in 12 computers (SuperMicro 6010H). Each machine contains two 1Ghz Pentium III processors, 512Mby RAM, 2 fast ethernet adapters and one 18Gbyte SCSI disk.
One of the machines, acting as a "master" node, distributes via a private network the MOSIX kernel and the operating system to the other machines, using the same diskless architecture shown in section 3**.**
Client nodes can be considered as diskless machines; their local disk is only used as a "scratch" area.
Each node is connected to a second private network to access applications and data, stored on a 144Gbyte SCSI disk array, attached to an Alphaserver 4100 (dual 500Mhz alpha processors + 256Mb RAM) and exported via standard NFS.
The following picture shows the entire architecture:

[3] *http://clusternfs.sourceforge.net*

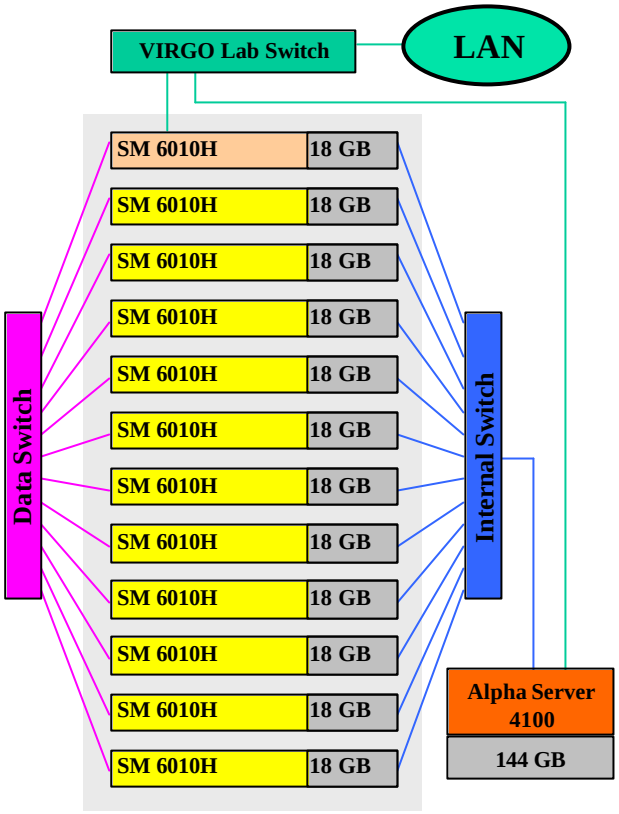

Figure 2: ***VIRGO farm architecture***

Each node currently runs Linux RedHat 7.2 and OpenMOSIX 2.4.17, while the AlphaServer runs Compaq Tru64 (5.0a).
Most of the applications running on the cluster use some math libraries such as Numerical Recipes and Grasp, a library for gravitational signal processing.
Some applications use parallel implementation of filtering algorithms for gravitational signals based on MPI.

One of the main purposes of the Linux farm in Naples is the detection of a particular class of gravitational waves; those emitted by coalescing binaries.
This type of analysis is performed using specific libraries, written by the Virgo researchers, and some public domain math tools, such as FFTW and ROOT.
The Linux farm has been strongly tested by executing intensive data analysis procedures, based on the Matched Filter algorithm, one of the best ways to search for known waveforms within a signal affected by background noise.

Matched Filter analysis requires a high computational cost as the method consists in an exhaustive comparison between the source signal and a set of known waveforms, called “templates”, to find possible matches. Using a large number of templates the quality of known signals identification gets better and better but a great amount of floating points operations has to be performed.
Running Matched Filter test procedures on the OpenMOSIX cluster have shown a progressive reduction of execution times, due to a high scalability of the computing nodes and an efficient dynamic load distribution.
Figure 3 shows the measured speed-up of repeated Matched Filter executions:

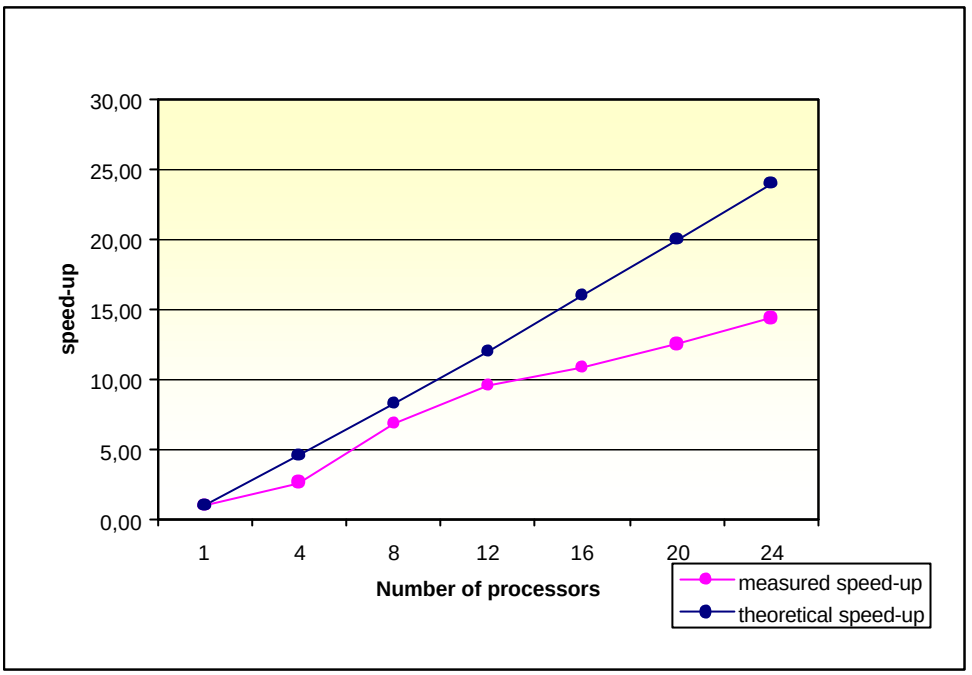

Figure 3: ***Speed-up of parallel execution of Matched Filter program***

The increase of computing speed respect to the number of processors doesn't follow an exactly linear curve; this is mainly due to the growth of communication time, spent by the computing nodes to transmit data over the local area network.

ARGO

The aim of the ARGO-YBJ [9] experiment is to study cosmic rays, mainly cosmic gamma-radiation, at an energy threshold of ~100 GeV, by means of the detection of small size air showers.
This goal will be achieved by operating a full coverage array in the Yangbajing Laboratory (Tibet, P.R. China) at 4300 m a.s.l. .

As we have seen for the Virgo experiment, the analysis of data produced by Argo requires a significant amount of computing power. To satisfy this requirement we decided to implement an OpenMOSIX cluster.
Currently Argo researchers are using a small Linux farm, located in Naples, constituted by five machines (dual 1Ghz Pentium 3 with a total amount of 4.5 Gbyte RAM) running RedHat 7.2 and Mosix 2.4.13.

The cluster has been set up using the same diskless architecture we have previously illustrated (Etherboot+ClusterNFS).
At this time the Argo OpenMOSIX farm is mainly used to run Monte Carlo simulations using *Corsika*[4], a Fortran application developed to simulate and analyse extensive air showers.
The farm is also used to run other applications such as *GEANT*[5] to simulate the behaviour of the Argo detector.
The OpenMOSIX farm is responding very well to the researchers' computing requirements and we already decided to upgrade the cluster in the near future, adding more computing nodes and starting the analysis of real data produced by Argo.

[4] *http://ik1au1.fzk.de/~heck/corsika/*
[5] *http://www.fisica.unile.it/~argo/analysis/argog/*

Conclusions

For our purposes, the most noticeable features of OpenMOSIX are its load-balancing and process migration algorithms, which implies that users don't need to have knowledge of the current state of computing nodes.
Parallel application can be executed by forking many processes, just like in an SMP, where OpenMOSIX continuously attempts to optimise the resource allocation.

The "(Open)MOSIX+EtherBoot+ClusterNFS" approach is one of the best solutions to build powerful computing farms with minimal effort.

In our environment, the choice of (open)Mosix has been proven to be an optimal solution to give a general performance boost on implemented systems.

References

[1] OpenMosix homepage
http://www.openmosix.org
[2] A. Barak, O. La'adan: "The MOSIX Multicomputer Operating System for High Performance Cluster Computing", Journal of Future Generation Computer Systems, Vol. 13, March 1998
http://www.mosix.cs.huji.ac.il/ftps/mosixhpcc.ps.gz
[3] Ken Yap, Markus Gutschke: "Etherboot User Manual", 2002
http://www.etherboot.org/doc/html/userman.html
[4] G. R. Warnes: "Recipe for a diskless MOSIX cluster using ClusterNFS", 2000
http://clusternfs.sourceforge.net/Recipe.pdf
[5] "General overview of the VIRGO Project"
http://www.virgo.infn.it/central.html
[6] F. Barone, L. Milano, R. Esposito et al.: "Evaluation of Mosix-Linux Farm Perforamces in GRID Environment", Proc. of CHEP (International Conference on Computing in High Energy and Nuclear Physics), pag. 702-703, September 2001
[7] F. Barone, L. Milano, R. Esposito et al.: "Mosix Linux Farm Prototype for GW Data Analysis", 6th Gravitational Wave Data Analysis Workshop, December 2001
http://gwdaw2001.science.unitn.it/abstracts/abstractsgwdaw.pdf
[8] F. Barone, L. Milano, R. Esposito et al.: "Preliminary tests on the Napoli farm prototype for coalescing binary analysis", VIR-NOT-NAP-1390-196, March 2002
[9] Argo experiment homepage
http://www1.na.infn.it/wsubnucl/cosm/argo/Argo_index.html