<u>HowTo</u>

Heterogeneous Clusters

Running ClusterKnoppix as a master node to a CHAOS drone army

CONTROL PAGE

Document Approvals

Approved for Publication:

Author Name:	Ian Latter 12 December 2003

Document Control

Document Name:	Heterogeneous Clusters; Running ClusterKnoppix as a master node to a CHAOS drone army
Document ID:	howto - heterogenous clusters.doc-Release-1.1(467)
Distribution:	Unrestricted Distribution
Status:	Release
Disk File:	C:\Documents and Settings_.NULL\Desktop\whitepaper\HowTo - Heterogenous Clusters.doc
Copyright:	Copyright 2003, Macquarie University

Version	Date	Release Information	Author/s
1.1	*12-Dec-03*	Release / Unrestricted Distribution	Ian Latter
1.0	*11-Dec-03*	Draft / Uncontrolled	Ian Latter

Distribution

Version	Release to
1.1	Public Release
1.0	Macquarie University, Moshe Bar, Bruce Knox, Wim Vandersmissen

Table of Contents

1 Overview

There is a somewhat conflicting interest between the security and manageability of a compact operating system, and the need for a user friendly and fully featured operating system, in the cluster environment.

Heterogeneous clusters provide the best of both worlds.

In this two-part HowTo, you will find details on how a heterogeneous cluster works, why it is advantageous, and instructions on building a heterogeneous cluster from the ClusterKnoppix and CHAOS Linux openMosix distributions.

2 Why you want a heterogeneous cluster

2.1 Where applications live

An operating system allows an application to be launched from storage media, into memory, and to execute through to completion (termination). openMosix, as a "Single System Image", provides the ability for an application to be launched from any cluster node, into memory, and to execute on any node in the cluster (as a virtual instance of the application). Any given virtual application instance is migrated to the node with the most available capacity or resources.

Note, that the "home node" (or "master node") is the only node that the application need exist on storage media. Each and every other node ("drone node") in the cluster could run a distribution that is as small as the Linux kernel itself.

2.2 Optimizing cluster administration

Most clusters deployed for either academic, research or even commercial purposes will be deployed for a pre-determined intent; a thesis, an experiment, an ongoing service or product – an objective. As such, most clusters are driven by some form of "cluster operator", some person or group who is charged with achieving that objective on (or perhaps facilitating service to) the cluster infrastructure.

This type of deployment allows for tuning of the cluster environment to suite the objective; namely, removing the un-needed cluster functionality that must be maintained in comprehensively complete drone nodes.

Looking at the problem from the technical perspective, managing a cluster of 30 nodes, where each node contains a full Linux distribution (including GUI) is not so different from managing a single node that contains a full Linux distribution. From the security or TCO perspectives, however, each additional non-essential software package that is stored on these drone nodes, adds both unnecessary risk, and unwanted cost.

Ideally, the cluster should consist of one or two home nodes, furnished with the complete software suite required to effectively execute objective tasks on the cluster, and be supplemented with drone nodes that contain only enough software to start the host hardware and to join the cluster effectively.

2.3 Not all distributions are the same

Heterogeneous cluster construction, allows an organization to take advantage of the feature differences that separate the relevant Linux distributions. In other words, by planning for the cluster, selecting the right distribution for the master node(s) and the right distribution for drone nodes, you will build the right cluster for your organization.

As with the greater Linux community, the openMosix Linux distributions differ widely. Distributions such as ClusterKnoppix, Dynebolic, LiveLinuxCD (Grendelsbane / Basilisk), Quantian and Sentinix are designed to be good master nodes, whereas distributions such as CHAOS, openMosixLOAF and PlumpOS are designed to be good drone nodes.

2.4 ClusterKnoppix Objectives

ClusterKnoppix is a derivative of Knoppix. Knoppix, based on Debian, is designed to be a complete Linux distribution, running the largest possible array of desktop user applications. Knoppix provides this facility without actually “installing” the operating system onto local storage media – running the entire distribution from CDROM. This combination is due to the original Knoppix objective; to allow ordinary not-so-technical computer users to try Linux as a PC desktop operating system, without destroying any existing operating system installation.

ClusterKnoppix aims to provide the same core features and software as Knoppix, but adds the openMosix clustering capabilities also.

Being a comprehensive Linux and openMosix distribution, ClusterKnoppix is probably one of the most popular “master node” capable distributions in the openMosix community. It is included in this HowTo, however, because it is the first master node capable distribution that has adopted a suite of security enhancements from the CHAOS distribution.

2.5 CHAOS Objectives

CHAOS is designed to be a compact Linux and openMosix distribution that is secure and highly distributable. CHAOS does not interfere with the host operating system, running entirely from RAM it also frees the CDROM boot media.

CHAOS is not intended to be fully-laden with application software; it is well suited as a minimized “drone node” distribution. Though, the biggest advantage of using CHAOS is its focus on security and deployment methodology.

CHAOS is the first Linux and openMosix distribution to add native network security to the cluster deployment. Every node in the cluster does its own packet filtering, and establishes IPSEC tunnels as required. This is performed transparently, requiring no user intervention.

While CHAOS is capable of PXE booting new nodes, CD booting will be used for this HowTo.

3 Building the cluster

3.1 What you are building

The cluster that you are about to build will contain one master node (of type ClusterKnoppix) and one drone node (of type CHAOS). The mixture of master versus drone nodes is entirely up to you. For continuity, it would be wise to build clusters of at least two master nodes.

Arbitrary addressing will be used (private addresses from RFC1918). And two examples will be provided, one where the local network provides automated address assignment (BOOTP or DHCP) and another where manual assignment is required.

3.2 Prerequisites

You will need two Pentium (i586) class, or better, computer systems. These systems must each be connected to an ethernet network and must be able to boot from their CDROM drives.

The minimum required software versions for this HowTo are;

- ClusterKnoppix: *clusterKNOPPIX_V3.3-2003-11-19-EN-cl*
- CHAOS: *CHAOS-v0.7*

See the references for links to web sites that contain these distributions. You will need one CD copy of each distribution.

Due to the techniques used in this HowTo, these systems do not need to be on the same LAN or VLAN (broadcast domain), but we will assume that they are for the purposes of our simulated addressing.

For a minimal installation time and effort, you should run a BOOTP or DHCP service on your network. If one exists, the cluster installation process is very fast indeed.

The simulated addressing that we will use for this demonstration is;

```
Network:          192.168.1.0
Netmask:          255.255.255.0
Default Gateway:  192.168.1.1
```

The hosts will be arranged as follows;

```
Master1:          192.168.1.11
Drone1:           192.168.1.21
```

This HowTo will not go into technical detail regarding the individual OS or security components. There is a good deal of existing documentation that can assist you with these enquiries.

3.3 ClusterKnoppix: Initializing the master node

The cluster topology does not require the “master node” to be the first node installed. As openMosix is a cluster of “peers”, any node can be installed first. However, for the purposes of this HowTo, we will start with ClusterKnoppix – our master node.

To start the ClusterKnoppix;

```
    1. Boot from the ClusterKnoppix CD
    2. At the boot: prompt, press <enter>
```

Allow the distribution time to load the operating system, interrogate your hardware and load the graphical user interface. Once the GUI has loaded, start a terminal shell (an Xterm), then become the super user;

```
    3. su -
```

If the local network segment does not have an automatic IP address allocator (via a BOOTP or DHCP service), then execute the following two commands to configure your local network interface;

```
    4. ifconfig eth0 192.168.1.11
    5. route add -net 0.0.0.0 gw 192.168.1.1
```

Once you have a working local network interface, you can initialize the openMosix subsystem, and host-based security;

```
    6. tyd -f init
```

And start this node as the first node in the cluster;

```
    7. tyd
```

At this stage, the master node should now be initialized and operating correctly.

3.4 CHAOS: Initializing drone nodes

CHAOS goes as far as it can to minimize boot process interaction in environments that support automatic IP address allocation.

To start CHAOS in an environment that supports automatic IP address allocation;

```
    1. Boot from the CHAOS CD
    2. At the boot: prompt, press <enter>
```

Allow the distribution time to load the operating system, interrogate your hardware and load the operating shell. Then, to start this node as a subsequent node in the cluster, execute;

```
    3. tyd –m 192.168.1.11
```

This is all that is required to add a new drone node in a network that contains either a BOOTP or DHCP automatic allocation service.

If, however, the environment requires manual address allocation, there are two methods for doing so.

The first method is to configure the ethernet interface from the OS boot prompt;

```
    1. Boot from the CHAOS CD
    2. At the boot: prompt, press <F4>
```

You will be presented with a help screen that instructs on IP address configuration options that can be specified before the boot process begins. Some may find this an easier way of interacting with the node addition process. The special kernel option "ip=" allows the operator to specify these host IP address parameters;

ip=<client-ip>:<server-ip>:<gw-ip>:<mask>:<name>:<dev>:<PROTO>

To boot this node as Drone1, you would enter the following, at the boot: prompt;

```
    3. /chaos/bzImage root=/dev/ram0 initrd=/chaos/chaos.rdz
       rw boot=/dev/cdrom:iso9660 ip=192.168.1.21::192.168.
       1.1:255.255.255.0:::
```

This will allow init to plumb and route the interface, establish an appropriate host name and initialize host security.

Allow the distribution time to load the operating system, interrogate your hardware and load the operating shell. The last step in the process, then, would be to start this node as a subsequent node in the cluster, by executing the following;

```
    4. tyd –m 192.168.1.11
```

This is all that is required to add a new drone node, if the interface is configured at the boot screen.

The second method is to configure the ethernet interface from the command line, as was done in ClusterKnoppix;

```
    1. Boot from the CHAOS CD
    2. At the boot: prompt, press <enter>
```

Allow the distribution time to load the operating system, interrogate your hardware and load the operating shell. Then execute the following three commands to configure your local network interface and establish the appropriate host name;

```
    3. ifconfig eth0 192.168.1.21
    4. route add -net 0.0.0.0 gw 192.168.1.1
    5. fix_hosts
```

Now you can initialize the openMosix subsystem, and host-based security;

```
    6. tyd -f init
```

And start this node as a subsequent node in the cluster;

```
    7. tyd –m 192.168.1.11
```

At this stage, the drone node should now be initialized and operating correctly.

3.5 CHAOS: Warning about certaintyd

Being heavily security focused, CHAOS contains a special application called "certaintyd" (the Certainty Daemon). Certaintyd operates by performing an md5 checksum on the CHAOS root file-system (/dev/ram0) every 2 seconds, 30 seconds after it has booted. If the memory file-system changes in any way, certaintyd considers this a loss of node integrity, killing init (process 1), causing the node to leave the cluster and reboot.

If you are going to spend more than 20 seconds working in the CHAOS shell, then kill certaintyd from memory, or you may get a rude surprise.

3.6 Determining cluster status

From your master node, you should be able to run any of the native status or monitoring commands to check on the current cluster status.

Mosmon provides fundamental node resource information;

```
    Mosmon
```

Mtop is the openMosix version of "top", which shows where processes are in the cluster and what CPU and memory resources they are consuming;

```
    mtop
```

Showmap will dump a text copy of the cluster node database;

```
    showmap
```

3.7 Where to go from here?

Adding drone nodes is straight forward. Simply repeat the documented process, adding each new node as required. Adding master/home nodes is equally simple, but note that;

```
    tyd
```

is only ever used on the first node in a new cluster. The first node is not necessarily a master node; it is just the first node. All subsequent nodes (masters or drones) are added with a command like;

```
    tyd –m <ip address>
```

Help and information can be obtained from tyd, by executing;

```
    tyd ––help
```

At the time of writing, neither tyd nor omdiscd supports node removal. Clusters of temporary nodes can get "dirty" over time.

4 Conclusion

This HowTo details the advantages and methods for constructing heterogeneous Linux openMosix clusters.

While much emphasis is placed on the two selected distributions, there is no reason why the enabling technologies (namely tyd) could not be ported to the remaining distributions, to provide even greater flexibility for those organizations wishing to deploy SSI technology securely.

It is hoped that this documentation has clarified at least one of the of the many deployment meals, available from the openMosix community menu.

5 References

5.1 Papers / Presentations

5.1.1 Security and openMosix; Securely deploying SSI cluster technology over untrusted networking infrastructure

http://itsecurity.mq.edu.au/papers/White Paper - Security and openMosix.pdf

Ian Latter, December 2003.

5.1.2 HPC Computing Applied to Business Applications

http://openmosix.sourceforge.net/Business_Applications_2003.pdf

Moshe Bar, 2003.

5.1.3 Turning a group of independent GNU/Linux workstations into an (Open)Mosix cluster in an untrusted networking environment and surviving the experience

http://www.democritos.it/events/openMosix/papers/cagliari.pdf

Giacomo Mulas, November 2002.

5.2 Distributions

5.2.1 CHAOS

http://itsecurity.mq.edu.au/chaos/

5.2.2 ClusterKnoppix

http://bofh.be/clusterknoppix/

5.3 Software

5.3.1 openMosix

http://www.openmosix.org/

6 Contact

6.1 Additions, Modifications and Deletions

For changes to this document, please refer to the author and revision history blocks in the control page. Please report errors or omissions to the author.

6.2 Consultation

If you would like to discuss SSI security architecture or other concepts related to this HowTo, then please contact the author;

Ian Latter

IT Security Officer

Macquarie University, Australia.

Email: Ian.Latter@mq.edu.au