

Linuxha.net Administrator's Reference

Version	Date	Author	Comments
Draft	1 st December 2003	S. Edwards	Work in progress - this covers version 0.4.0 of the "Linuxha" code.
0.4.1	1 st January 2004	S. Edwards	Improvements made after installation testing of the 0.4.0 code base and changes to reflect 0.4.1 software.
0.5.0	24 th February 2004	S. Edwards	Updates to documentation to support changes between 0.4.0 and 0.5.0.
0.6.0	20 th April 2004	S. Edwards	Updates to include changes and features introduced for version 0.6.0. Title and style changed.
0.6.2	28 th June 2004	S. Edwards	Interim release for initial port to Linux 2.6.
0.6.4	22 nd July 2004	S. Edwards	Yet more changes reading for 0.7.0 release of software.
0.7.0	2 nd September 2004	S. Edwards	Further changes to detail features introduced in version 0.7.0
0.8.0	Janary 2005	S. Edwards	Guess what? Further program improvements are documented, as well as improved documentation for existing features.
0.9.0	10 th April 2005	S. Edwards	Final version covering pre-release of 1.0.0.

Table of Contents

1	Preface.....	9
1.1	Introduction.....	9
1.2	Document Organisation.....	9
1.3	Intended Audience.....	10
1.4	Conventions.....	10
1.5	Request for Feedback.....	11
1.6	Software Versions Covered.....	11
1.7	Understanding Linuxha.net Releases.....	11
2	Principals of High Availability.....	14
2.1	Redundancy.....	14
2.2	Availability.....	14
2.3	Client / Server Architecture.....	15
2.4	Linuxha.net Resource Limits.....	17
3	Managing Data Redundancy.....	18
3.1	Shared Storage Architecture.....	18
3.2	Replicated Storage Architecture.....	20
4	Data Replication with Linuxha.net.....	22
4.1	Choosing the Network Block Device.....	22
4.2	Using DRBD on the Command Line.....	23
4.3	Scenarios Requiring Resynchronisation.....	24
4.4	File System Support.....	25
5	Linuxha.net Solution Architecture.....	27
5.1	Definitions of Terms.....	27
5.2	Application Storage Management.....	28
5.3	Managing Cluster Consistency Status Information.....	30
5.3.1	Handling Specific Limitations.....	31
5.4	Cluster Status Daemon.....	31
5.5	Cluster Lock Daemon.....	32
5.6	Cluster Network Daemon.....	32
5.7	Application Monitoring with "Lems".....	34
5.7.1	System Monitors.....	34
5.7.2	Application Monitors.....	34
5.8	Cluster Utilities.....	36
5.9	Third Party Software.....	37
6	Installing Linuxha.net.....	39
6.1	Hardware requirements.....	39
6.2	Typical Hardware Configurations.....	39
6.2.1	Configuration 1 - No redundancy.....	40
6.2.2	Configuration 2 - Basic network redundancy.....	42
6.2.3	Configuration 3 - Local Storage Redundancy.....	43
6.2.4	Configuration 4 - Complete network redundancy (Multi-pathing).....	43
6.2.5	Configuration 5 - Multiple Public Networks.....	44
6.2.6	Specific Hardware Concerns.....	45
6.2.7	Hardware Configuration Conclusions.....	45
6.3	Environment Configuration.....	47
6.4	Kernel Configuration.....	48
6.5	Prerequisite Software.....	48
6.5.1	Installing Tarp Package Management Software.....	49
6.6	Installation of the "linuxha" package.....	51
6.6.1	Manual Check for LVM Support.....	52
6.6.2	Update the Superuser's PATH.....	52
6.7	Installation differences between 2.4 and 2.6 kernels.....	53
6.8	Upgrading from previous Installations.....	53
7	Building the Cluster Configuration.....	55
7.1	Detailed Example Cluster Configuration.....	55
7.2	Initial Cluster Build.....	58
7.2.1	Field-by-Field explanation of "clconf.xml".....	60
7.2.2	Heartbeat Considerations.....	64
7.2.3	Cluster Network and Locking Support.....	66

7.2.4	Building the Cluster.....	67
7.2.5	Creating "resource" flags.....	69
7.2.6	Running the "clbuild" Utility.....	70
8	Building the sample "apache" package.....	72
8.1	Creating the Application Configuration File.....	72
8.2	Network Availability Considerations.....	76
8.2.1	Bonding verses Fail-over network Types.....	76
8.2.2	Supported Bonding Modes.....	78
8.2.3	Link-level checking verses IP level checking.....	78
8.2.4	Sharing Multiple IP addresses.....	79
8.3	Checking the Application Configuration.....	79
8.4	Validation / Build of Volume Groups.....	82
8.5	Allocating Application Resources.....	83
8.6	Synchronising the Cluster File systems.....	84
8.6.1	Managing Application File systems "outside" the cluster.....	85
8.7	Understanding Application Resources.....	86
9	Application Configuration and Monitoring.....	88
9.1	Start/Stop Script Interface Requirements.....	89
9.2	Providing a "Lems" Monitor for the application.....	90
9.3	Sample Process Monitor Implementation.....	91
10	Starting the Cluster.....	94
10.1	Forming a Cluster using "cldaemon".....	94
10.1.1	When to use force to form a cluster.....	97
10.2	How a Cluster is Joined.....	97
10.3	Forming a Cluster using "clform".....	97
10.4	Forming a Cluster on machine boot.....	98
11	Managing Applications in the Cluster.....	101
11.1	Starting Applications with "clstartapp".....	101
11.2	Some typical Error Conditions when Starting Applications.....	102
11.3	Checking Application Status.....	105
11.3.1	The "File Systems" information.....	106
11.3.2	The "Process Monitors" information.....	107
11.3.3	The "General Monitors" information.....	107
11.4	Stopping Applications.....	108
11.5	Starting Applications (the easy way).....	110
11.6	Managing application Monitoring.....	111
12	Application Removal.....	113
13	Stopping the Cluster.....	116
13.1	Stopping the Cluster Manually.....	116
13.2	Stopping the Cluster Automatically.....	116
13.3	Halting Individual Nodes.....	117
13.4	Adding Nodes to a Running Cluster.....	118
14	Adding further Applications.....	119
14.1	Purpose of this section.....	119
14.2	Application Storage Requirements.....	119
14.3	Application Configuration for cluster.....	121
14.4	Start and Stop scripts for "Samba" Application.....	123
14.5	User Environment.....	123
14.6	Cluster Configuration.....	124
14.7	Checking the new Application Configuration.....	126
14.8	Allocating Application Resources.....	127
14.9	Limitations with Samba Sample Application.....	130
14.10	Adding Applications: Common Problems.....	130
14.10.1	Mismatch Security Settings.....	131
14.10.2	Missing Mount Points.....	131
15	System Administrator Responsibilities.....	133
16	Managing Application Monitoring.....	136
16.1	Stopping Monitoring.....	136
16.2	Resuming Monitoring.....	138
16.3	Pausing a Module.....	139
16.4	Resuming a Module.....	139
16.5	Removing a Monitor.....	140

16.6	Adding a new Monitor.....	141
16.7	Monitor Specific Communication.....	142
16.8	Log Management.....	143
16.8.1	Handling of Compressed Logs.....	144
16.9	Stopping and Starting the Lems Daemon Manually.....	145
17	Managing Cluster Daemons with "cldaemonctl".....	146
17.1	Log File management.....	146
17.1.1	Application Specific Logs.....	146
17.1.2	Clstartapp and Clhaltapp specific Logs.....	147
17.1.3	Cluster daemon log files.....	147
17.2	Resetting Application Fail-over Capability.....	148
17.3	Stopping Application Fail-over Capability.....	149
17.4	Checking Cluster Status in a Script.....	149
17.5	Starting and Stopping Applications.....	150
18	Managing Configured Applications.....	152
18.1	Adding new file systems.....	152
18.1.1	Online Addition of file systems.....	152
18.1.2	Offline Addition of file systems.....	154
18.2	Removing existing file systems.....	155
18.2.1	Online Removal of File systems.....	155
18.2.2	Offline Removal of File systems.....	156
18.3	Changing existing file systems.....	156
18.3.1	Online file system expansion.....	157
18.3.2	Offline file system expansion.....	157
18.4	Modification of application parameters.....	158
19	Easier Application Management.....	159
19.1	Differences During Application Build.....	160
19.2	Using the "clrunapp" Utility.....	160
19.3	Using the "clform" Utility.....	162
19.3.1	Joining an Existing Cluster.....	163
20	Performing Software Upgrades.....	165
20.1	Background Information.....	165
20.2	Upgrading Clustered Applications.....	165
20.3	Upgrading Linuxha.net Software.....	167
20.4	Updating Operating System Software.....	171
21	Handling Failure Scenarios.....	173
21.1	Introduction.....	173
21.2	Common Failure Scenarios.....	173
21.2.1	Loss of a Network Link.....	174
21.2.2	Handling IP-level failures.....	177
21.2.3	Failure of Data Replication Network Connection.....	177
21.2.4	Application Software Failure.....	178
21.2.5	Stopping Fail-over (from application monitoring).....	179
21.2.6	Process Monitor Administration.....	181
21.2.7	Node failure (Hardware or Operating System).....	181
21.3	Managing Node Failure Scenarios.....	182
21.3.1	Checking Application Status.....	182
21.3.2	Recovering Application Data Availability.....	184
21.4	Loss of Server Main IP Interface.....	186
21.5	Loss of Cluster Daemon.....	188
21.6	Understanding Network Partitioning.....	189
21.7	Data Consistency Issues.....	191
22	Implementation details of "clstartapp" & "clhaltapp".....	193
22.1	Supported Command Line Arguments.....	193
22.2	Default Argument Settings.....	194
22.3	Ascertaining Cluster Status.....	195
22.4	Starting cluster packages - Condition Decisions.....	195
22.4.1	Actions on a Clean Start-up.....	198
22.4.2	Actions on a Clean Shutdown.....	199
22.4.3	Actions on a non-Clean Shutdown (no remote available).....	199
22.4.4	The "getmdlist" Utility.....	201
23	Implementation Details of "clrunapp".....	202

23.1	Introduction.....	202
24	Understanding the "Lems" Daemon.....	204
24.1	Introduction.....	204
24.2	The Lems Configuration File.....	204
24.3	The Lems Action List.....	208
24.3.1	Using the "RUNCMD" Action.....	210
24.4	Standard "Lems" Monitors.....	211
24.4.1	The "Flag_check" Module.....	212
24.4.2	The "Ip_module" Module.....	212
24.4.3	The "Link_module" Module.....	212
24.4.4	The "Ip_move_interface" Module.....	212
24.4.5	The "capacity_check" Module.....	213
24.4.6	The "swap_check" Module.....	213
24.4.7	The "Fsmon" Module.....	213
24.4.8	The "procmon" Modules.....	217
24.5	The "Lems" Server Messages.....	218
24.6	The "Lems" Module Object Method Requirements.....	220
24.6.1	Object Environment.....	221
24.7	The "Lems" Program Execution Requirements.....	221
24.8	Writing a Sample Lems Monitor.....	221
24.8.1	The "new" method.....	221
24.8.2	The "check" Method.....	223
24.8.3	The "stat" Method.....	223
25	Understanding the Cluster Management Daemon.....	224
25.1	Introduction.....	224
25.2	How the Cluster Daemon Interacts with an Application.....	224
25.3	Forming the Cluster - Cluster Daemon Initialisation.....	225
25.4	Protocol and Messages.....	227
26	Cluster Utility Scripts.....	235
26.1	Starting "nbd" or "enbd" Servers.....	235
26.2	Starting "enbd" Clients.....	236
26.3	Stopping "nbd" or "enbd" Clients.....	236
26.4	Generating "Raidtab" Configuration Files dynamically.....	236
1	Cluster Resource Management.....	239
26.5	Application IP Address Management.....	239
26.6	NBD / ENBD Server Management.....	239
27	Kernel / System Software Configuration.....	240
27.1	General Configuration.....	240
27.2	The "Raid" and "LVM" Modules.....	240
27.3	The "bonding" Module.....	241
28	Custom Perl Modules.....	242
28.1	The "fsmap" Module.....	242
28.2	The "clutils" Module.....	244
28.3	The "clbonding" Module.....	244
29	Application Directories.....	247
29.1	Non-Standard Perl Packages.....	247
A.	Understanding "ENBD".....	249
A.	Understanding "ENBD".....	249
A.	Understanding "ENBD".....	249
I.	Introduction.....	249
I.	Introduction.....	249
I.	Introduction.....	249
II.	An "Enbd" client / server Example Walkthrough.....	249
II.	An "Enbd" client / server Example Walkthrough.....	249
II.	An "Enbd" client / server Example Walkthrough.....	249
III.	Stopping the Client.....	250
III.	Stopping the Client.....	250
III.	Stopping the Client.....	250
IV.	Understanding Device Resource Allocation.....	251
IV.	Understanding Device Resource Allocation.....	251
IV.	Understanding Device Resource Allocation.....	251
B.	Using Raid Modules for Data Replication.....	253

B.Using Raid Modules for Data Replication.....	253
B.Using Raid Modules for Data Replication.....	253
I.Introduction.....	253
I.Introduction.....	253
I.Introduction.....	253
II.Basic Raid Operations.....	253
II.Basic Raid Operations.....	253
II.Basic Raid Operations.....	253
III.Understanding /proc/mdstat.....	253
III.Understanding /proc/mdstat.....	253
III.Understanding /proc/mdstat.....	253
IV.Synchronising Multiple Devices.....	255
IV.Synchronising Multiple Devices.....	255
IV.Synchronising Multiple Devices.....	255
C.Setting up SSH.....	257
C.Setting up SSH.....	257
C.Setting up SSH.....	257
I.Introduction.....	257
I.Introduction.....	257
I.Introduction.....	257
II.Step 1: Define Public/Private Key Pairs.....	257
II.Step 1: Define Public/Private Key Pairs.....	257
II.Step 1: Define Public/Private Key Pairs.....	257
III.Defining Machine Definitions.....	258
III.Defining Machine Definitions.....	258
III.Defining Machine Definitions.....	258
IV.Define SSH client authorization.....	258
IV.Define SSH client authorization.....	258
IV.Define SSH client authorization.....	258
D.Guidelines for Editing XML Files.....	260
D.Guidelines for Editing XML Files.....	260
D.Guidelines for Editing XML Files.....	260
E.Sample "Apache" Application Configuration.....	261
E.Sample "Apache" Application Configuration.....	261
E.Sample "Apache" Application Configuration.....	261
F.Introduction.....	261
F.Introduction.....	261
F.Introduction.....	261
G.Building the Volume Group.....	261
G.Building the Volume Group.....	261
G.Building the Volume Group.....	261
H.Using the Sample files.....	264
H.Using the Sample files.....	264
H.Using the Sample files.....	264

1 Preface

1.1 Introduction

The purpose of this administration guide is manifold;

- to describe the typical installation process;
- to define the support environment requirements;
- define how a cluster can be built;
- define how applications can be added to the cluster;
- cover the typical cluster management tasks that may be necessary;
- describe typical failure and recovery scenarios;
- provide a technical reference for developers.

This document is a significant update from previous versions; where possible references to other "guides" that have been produced will be indicated. This document is now wholly aimed at being the "reference" to Linuxha.net - the series of "guides" provide less in-depth material aimed at documenting particular requirements. Hence both this document and the guides complement one another.

This guide also attempts to introduce the general principals that guide most high availability solutions, though of course all content will be geared towards solutions that Linuxha.net could be architected for.

It should be noted that high availability although simple in principal is much more complex when implemented, particularly when availability via data replication (which is the basis of of Linuxha.net). The key aim of Linuxha.net is to hide as much complexity as possible whilst providing a powerful environment for clustered application deployment.

The Linuxha.net product has a dedicated web site where documentation, news and latest versions of the software can be retrieved from:

<http://www.linuxha.net//index.pl?ARGS=findproject:linuxha>

Prior to installation of any version of Linuxha.net it is recommended that the latest news items are read to check for any new releases notes that may impact installation or use of the software.

1.2 Document Organisation

This reference is broken down into four main parts:

Part 1 – *Introduction to Data Replication Clustering* – gives an overview of the design considerations that Linuxha.net uses and attempts to describe some of the problems a replicated data cluster some overcome.

Part 2 – *Building a sample cluster* – this describes the installation and configuration of the software providing a cluster - including installation of a sample application.

Part 3 – *Day-to-day cluster administration* – The previous section contains probably almost all the information needed to manage the cluster assuming everything is going well. This section goes into more details of long term cluster management - such as adding, changing or removing applications - even when the cluster remains up and running.

Part 4 - *Trouble shooting guide* - A section dedicated to describing common failure scenarios along with information on how Linuxha.net handles these situations and details of any manual intervention that may be required to complete

Part 5 – *Technical information for the curious* – This section contains copious technical details describing the workings of the cluster software. Only relevant to those who intend to customise the software or contribute to the project.

1.3 Intended Audience

This document is intended to be read by those responsible for either implementing a new Linuxha.net cluster, performing day-to-day cluster management, or anyone attempting to recover from a failure scenario which is not dealt with automatically.

The text does have some expectations of the reader; namely a reasonable experience of Linux or other UNIX environment; comfort with the command line interface, a basic knowledge of XML and use of a standard editor (such as "vi") to alter configuration files.

Since Linuxha.net makes use of the Linux Logical Volume Manager it is recommended that the reader become familiar with the basic concepts and commands before attempting to use this software.

1.4 Conventions

Any commands that can be entered on the command line will appear in `courier` font with a light grey background, for example:

```
# clstat
```

Any commands that must be entered as the "root" user will be prefixed with a "#" symbol, whilst commands entered by other users are prefixed with "\$", for example:

```
$ ls
```

Any output is typically shown in the same format, and may immediately follow the command or appear after an explanation - though does not include the background colour, for example:

```
# clstat
Cluster: cluster2 - DOWN
```

Any points that is of particular interest it will be shown as a note, highlighted as follows:

This is a important consideration of which you should be aware.

If a point is considered critically important it will appear as follows:

If you ignore this you might loose all you data.

Finally if a particular point is generally considered useful, but not highly critical it will be denoted as follows:

This is a useful tip which might make your Linuxha.net experience more manageable.

1.5 Request for Feedback

This documentation and the associated software is available for free via the GNU GPL (software) or Creative Commons (Documentation) licenses. However it would be a great benefit if anyone is able to send details of their experiences (good or bad) to the following address:

simon.edwards@linuxha.net

All feedback is confidential and will be used purely to help improve the documentation and software.

1.6 Software Versions Covered

This document covers any "pre-release" version of the software from 0.9.0 through to 0.9.9. When a 1.0.0 release is made available this document will be updated accordingly - though it is

likely that the changes will be only for fixes and new material since no new "features" are being added between the versions 0.9.0 and 1.0.0.

1.7 Understanding Linuxha.net Releases

Due to the limit resources it is not possible to test every release of Linuxha.net against every supported configuration prior to the release of 1.0.0. However certain releases are tested more thoroughly than others:

- x.x.0 The ".0" releases, such as 0.9.0 tend to be tested the most. Although this does not guarantee that all functionality across all operating system versions has been tested, it does mean that such versions are less likely to suffer from unknown bugs or lack of functionality.
- x.x.n These "minor" releases are used to fix bugs or problems either currently outstanding from prior to the last major release, or newly found problems with the last major release. The intention is to expose this new functionality to environments for as much testing as possible - and hence installation of such releases should be performed with caution.

Once 1.0.0 is released the mechanism will change:

- 1.0.x These releases will be bug-fixed only. Each release will under go a complete end-to-end test to ensure no new bugs are introduced.
- 1.1.x A development tree - will include new features and also bug fixes from the 1.0.x releases when appropriate. None of the 1.2.x should be considered for production environments.
- 1.2.x Stable releases taken from the 1.1.x code-base. Each will be tested using the end-to-end test suite that will be deemed appropriate for the 1.2.x releases.

Hence 1.1.x and 1.2.x will share the same source code, whilst 1.0.x will continue to use the original source code repository which is used for this document.

It is intended that the 1.0.x releases are backwardly compatible with 0.9.0 and above, whilst this will not be true for the 1.1.x and 1.2.x releases.

Part I:
An introduction to Replicated Data Clustering

2 Principals of High Availability

Before it is possible to design a high availability solution for any application some of the basic principals involved in high availability must be understood. Given the simplistic goal of the Linuxha.net product, the two main concepts of concern are “redundancy” and “availability”. However typically highly available environments are client/server architectures, where the “server” side is that which is highly available.

For example consider a highly available Samba application - Linuxha.net could be used to ensure the server component was made available, but does nothing to guarantee the other infrastructure components such as the clients, networks, or shared printers.

2.1 Redundancy

Redundancy simply means that there should be more than one component in a system capable of doing the same job. The list below gives you an example of typically which components are made redundant in a highly available environment:

- [1] More than one network card
- [2] More than one copy of the data
- [3] More than one machine
- [4] More than one location
- [5] More than one power source
- [6] More than one network

Of course it is usually beyond most budgets to afford all these in a solution. However to build a high availability environment at least the first three must be met. The minimum requirements for Linuxha.net are [2] and [3], whilst it is able to improve availability as more of the conditions are met.

From this point onwards the term “cluster” will be used to describe the collective hardware used to create the highly available environment. A single machine is referred to interchangeably as a “server” or “node”. The Linuxha.net software supports two “node” clusters only - though support for 3+ nodes will be added for future releases.

2.2 Availability

A highly available system is not one without *any* downtime – it merely means that under most circumstances if there is an outage the affected applications will be made available again to any clients within a designated time period, (typically in less than 1 minute, but may be more depending on the complexity of the application - starting Oracle's iAS products can take many minutes to start, for example).

Note that this means that the failure of the application is usually not transparent – the user typically has to reconnect to the service in question, though much does depend on the intelligence of the client/server architecture of the application in question.

Again, using Samba as an example, no reconnection will be necessary - the shares appear to “unfreeze” once the fail-over of the application to another node is complete. Actually Linuxha.net provides availability at several levels:

[1] Application Availability

Tools are provided to monitor that an application is running, and if not to re-start locally, or to restart it on the other node if the software appears to fail too frequently on the current node.

[2] Network Availability

The cluster topology contains information on various networks supported by the cluster. The cluster is then able to monitor physical network connectivity and fail-over IP connectivity from one network interface to another if deemed necessary. Such fail-overs are sub-second and are usually completely transparent.

[3] Host Availability

If a node appears to die then the software will re-start any applications that were running on that node at the time - assuming that the remaining node was up and running of course!

2.3 Client / Server Architecture

Usually a high availability cluster consists of one or more applications acting as “services” to clients – be it a web server, database or file server for example. The common theme here is that there is the concept of client / server topology – so any application that can fall into such a category could (in theory) be placed in a cluster.

The Linuxha.net cluster software covered in this reference allows one or more programs to be made available. Each service that acts as a separate entity is known as an “application”. You may have zero or more applications in your cluster.

Running the cluster with no applications can actually be useful - in this case the network topology is still being monitored and thus IP fail-over when an interface problem becomes apparent still takes place.

The cluster can be run in two different “modes” (though these are just determined by how the administrator configures the applications and uses the commands, rather than actually being required to configure the cluster in a different manner). These modes are:

➤ Idle Standby

In this scenario all applications started on a single server. The other server in the cluster hosts no applications under normal circumstances - hence being classed as “idle”.

The advantage of this approach is that it is typically more straightforward to manage - it allows the “idle” node to be easily changed without impacting any running applications. Further since the node is typically not used it is possible to define it as a lower specification server (less memory, CPU's), thus reducing the overall hardware costs for the cluster.

Of course the major disadvantage of this method of working is hardware utilisation - a server is not performing useful work (from the perspective of client software). Thus even though the standby machine might have cost less, it is still essentially providing a low Return on Investment.

Idle standby is implicit if only a single application is being hosted by the cluster.

➤ Active Standby

In this scenario the clusters hosts two or more applications. Under normal operating conditions both servers run clustered applications, (though not the same applications).

Since both servers are running applications best use of the available hardware is being made. Of course this does make administration potentially more complex since the administrator can not assume where applications reside. For example if the administrator uses a script to pass information from the web server to the database server it is not possible to assume that they are running on the same server.

Because under these operating conditions both servers are critical physical loss of a server will result in some applications being migrated. With an idle standby there is only a 50% chance of the loss of the server causing any problems for client access!

Hence the approach taken is entirely determined by the requirements for each installation. Since the parameters of the running cluster and applications can be changed at any time as conditions dictate.

2.4 Linuxha.net Resource Limits

Many clustered environments typically limit certain resources. With Linuxha.net care has been taken to remove such limits where possible. The following table might be useful when comparing clustering products, commercial or otherwise.

Resource	Maximum Supported
Nodes	2 ¹
Applications	Unlimited
File Systems	256 - 2.4 Kernels Unlimited ² - 2.6 Kernels
File System Size	2 Tb
Networks	Unlimited
Cards per Network	Unlimited
IP's per Network	Unlimited
IP's per Application	Unlimited
Networks per Application	Unlimited

Notes from the above table:

1. Later versions of the software may include support for more nodes, but any application will still only be able to fail-over to one other designated node.
2. The number of file systems is determined by the available number of minor numbers for a device. For Linux 2.6 based kernels this means 65536 - which in terms of a number of file systems must be considered "unlimited".

3 Managing Data Redundancy

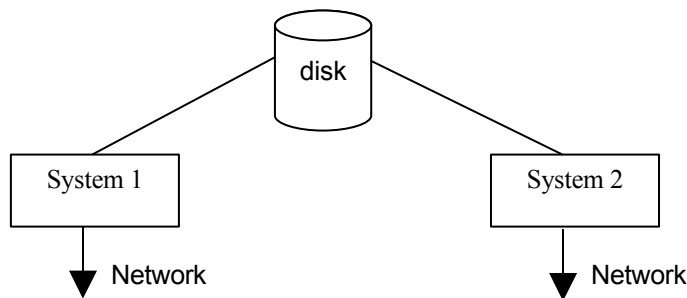
Linuxha.net uses a special kernel device driver called "DRBD" (see <http://www.drbd.org> for details). This driver allows data to be replicated to a remote server over a standard IP connection. Thus the data that the cluster uses is always local to the node using a particular application, with any changes being sent to the remote node for replication.

Data Availability is usually the keystone to making an application highly available. Few applications do not have any data (whether state, reference or log files), thus the method chosen to ensure the data is available to both nodes is critically important.

To understand the disadvantages and advantages of a replicated data approach it is useful to understand the alternative approach which most commercial clustering software makes use of - Shared Storage Architectures.

3.1 Shared Storage Architecture

This is the topology used by most commercially available clustering software - definitely all that allow more than two nodes (from my experience). In this case the storage used for the application data is stored on a device that is independent of either of the hosts in the cluster. The diagram below uses a typical topology used commercially today for many clusters:



Architecture

Figure 1: Shared Storage

The advantage of this architecture is that is relatively simple to configure and manage – there is only a single “copy” of the data, so thus there can only be one view of the data from which ever node you are running.

Since there is only a single data copy the number of nodes that can be attached to the data can be easily increased without significant complexity.

The major disadvantage of this approach is that the storage that is shared is in most cases likely to be hosted in a single cabinet, meaning loss of the power supply to the cabinet, or loss of the cabinet would result in the cluster becoming unavailable.

Of course this single point of failure (single data copy) can be overcome with mirroring of some type. Typical commercial environments are likely to use host-based mirrored or even dedicated SAN hardware related infrastructure - such as EMC's SRDF.

One other disadvantage of a shared storage architecture is cost - currently only SCSI or Fibre-channel based storage can be connected to multiple hosts - and such disks cost at least twice as much as existing PATA or SATA-based solutions.

3.2 Replicated Storage Architecture

The other typical choice for clusters is to use per-node local data copies - this topology is often referred to as a “replicated storage” topology. The diagram below demonstrates such an environment.

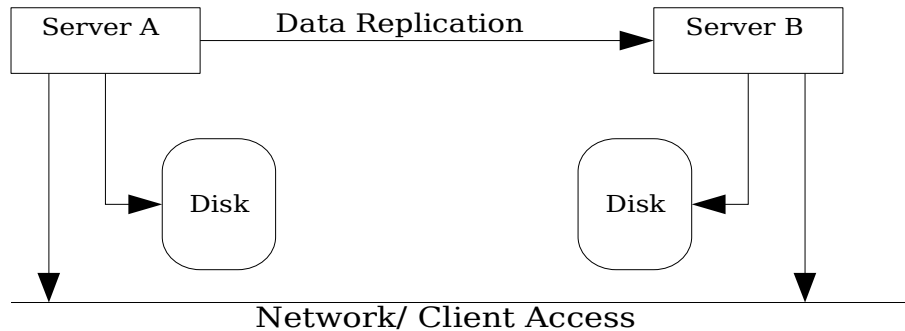


Figure 2: Replicated Storage Architecture

In this instance both machines keep a “local” copy of the application data, and uses this copy when the node is serving the application to clients. This immediately means that the cluster can only run an application on a single node at any one time - which is the solution typically required for clustering generic applications.

In this scenario when an application is using “Server A” is serving the application to clients it will also be sending any updates for the file systems used for that Application to “Server B”.

Hence in some respects even when running the application as an “idle standby” configuration the standby node is still performing work - ensuring its copy of the data remains up to date.

However this is an important distinction with replicated solutions – even if a node is not running an application it is not truly idle since it will be performing read/write activities for the application to. This must be taken into account when taking an “idle” node out of service.

In such instances the data replication infrastructure can be very flexible. Consider that the “local” storage does not necessarily need to be SCSI, PATA or SATA storage - it could be a Fibre Channel or iSCSI-based.

The key for a replicated storage solution obviously is a reliable connection to host the data to replicate. Linuxha.net supports replication over Ethernet hardware only - though the bandwidth should be carefully considered for the expected write performance of all running applications. The software will function using anything from 10Mb/sec half duplex through to 10Gb/sec full-duplex without problems.

In most circumstances if the servers are in the same location a single cross-over cable approach is possible, making a highly cost effective solution. Higher end approaches using channel bonding of Gigabit Ethernet connections are possible, if this is required.

The cost effectiveness of typical replicated solutions is one of the main advantages of this architecture. It is quite possible for provide 1Tb of replicated storage for a fraction of the cost of a shared storage solution.

A further advantage is that the machines can be in different locations, offering site resilience - a big benefit when considering disaster recovery requirements.

The major disadvantage of this approach is complexity – there are potentially two copies of the data – and due to the lack of atomic file systems as well as two copies it can be sometimes difficult to understand which copy is current and which is not.

Fortunately the DRBD device driver used is clever enough to transparently cope with most scenarios automatically. For example consider adding the standby node back into the cluster after a failure - which replicated file systems need updating? In such cases DRBD is able to keep track of differences and only update the portions of storage that have changed - a great benefit when replicating 100GB or more data.

4 Data Replication with Linuxha.net

The primary goal of Linuxha.net is to provide an administrator the ability to build a very cost-effective, highly available cluster - making use of data replication. This means that commodity hardware can be used (such as PATA or SATA drives) - allowing a cluster to be built for much less than might be possible using shared storage infrastructures.

This section now describes how the data replication is used within Linuxha.net - to provide resilience, but also to make things easier to use and administer for the system administrator.

4.1 Choosing the Network Block Device

To provide data replication over a IP link requires the use of a "network block device" - a device that uses the IP stack for communication to a remote service, whilst providing a standard "block" interface to the disk interface to make use of.

At present there are three well known "block device drivers" for Linux:

- **enbd** (formally "nbd")
This is based on the original "nbd" device driver and provides a minimum amount of functionality. To provide replication to a remote device it is necessary to layer this device as one side of a RAID-1 meta device, and use the RAID-1 device as storage.

This solution is known to offer good performance, but suffers from increased complexity and lacks sophisticated change-set handling in the event of local or remote failure. This was the original device used by earlier versions of Linuxha.net.

- **Grbd**
This device includes network block device support but is closely tied in with GFS (Global file system). Use of this outside of GFS networks does not appear to be possible.
- **Drbd**
This is the solution used currently by Linuxha.net. It addresses the recovery and complexity issues that plague Enbd and even provides several protocols which allow improved support for networks with high throughputs and/or high latencies.

The protocol that Linuxha.net uses by default is the safest - attempting to guarantee is best as possible remote data consistency. However this can be modified if necessary (though it is not recommended).

Because of the use of the "safe" protocol (known as "protocol C" in the DRBD camp), performance is not necessarily as good as Enbd using the same hardware and network infrastructure.

Architecture of DRBD-based replication

The diagram below shows how DRBD is used to replicate data.

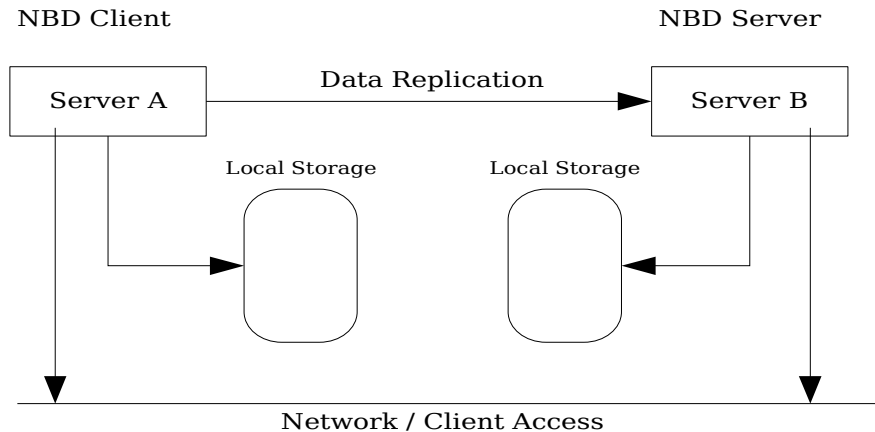


Figure 3: Using “DRBD” to replicate data

In the above diagram it is expected that the application using the data replication is currently running on “Server A”, replicating changes to “Server B”. Note:

- There are no explicit user-land processes running to provide what is essentially a client-server infrastructure (where the “client” in this case is actually the server than clients using the clustered applications connect to!).
- “Server A” is known as the “Primary” node - that is the DRBD device is open and in use, whilst “Server B” is known as the “Secondary” node - it’s DRBD is not in use, just being updated.
- If either one of the local or remote storage devices becomes unavailable the DRBD device on the primary will continue to function.
- Duration data resynchronisation operations, (such as when a failed node rejoins the cluster), DRBD uses the concept of a “SyncSource” and “SyncTarget”. The “SyncSource” is the node with the up to date information, hence pushing the changed data to the “SyncTarget”.

4.2 Using DRBD on the Command Line

The current version of Linuxha.net makes use of version 0.7.8 of DRBD. Although how this software is used internally within the product is not necessary to know understanding the mechanics of a DRBD device is worthwhile (your data depends on it).

The steps necessary to use this block device to mount a file system on “Server A”, are not particularly complex (once the initial data synchronisation has been completed): Firstly on “Server A”, run the following command:

```
Servera# drbdsetup /dev/drbd0 disk /dev/hda1
Servera# drbdsetup /dev/drbd0 net 175.100.0.1:9000 175.100.0.2:9000
```

The above commands device "drbd0" to use the local "/dev/hda1", whilst the remote connection is via "175.100.0.2" on port 9000 - (using "175.100.0.1" on port 9000) as the local TCP/IP connection.

Now similar commands must be executed on "Server B":

```
Serverb# drbdsetup /dev/drbd0 disk /dev/hdb2
Serverb# drbdsetup /dev/drbd0 net 175.100.0.2:9000 175.100.0.1:9000
```

These commands are the same as "Server A", though in this instance a different disk device is being used. At this point the devices will communicate, connect, and resynchronized automatically if necessary.

Before a device can be used, however it must be made "primary". To do this the following command should be run on "Server A":

```
Servera# drbdsetup /dev/drbd0 primary
```

At this point the "/dev/drbd0" can be used as a normal block device on "Server A", for example:

```
Servera# mount -t ffs /dev/drbd0 /apache
```

Checking Data Consistency

Currently the only method of checking the status of each used device by examining the contents of the "/proc/drbd" file. Hence consider the following command and output:

```
Servera# cat /proc/drbd
version: 0.7.10 (api:77/proto:74)
SVN Revision: 1743 build by root@sl4s2, 2005-05-16 22:56:13
 0: cs:Connected st:Primary/Secondary ld:Consistent
    ns:10 nr:0 dr:10 dr:236 al:2 bm:0 lo:0 pe:0 ua:0 ap:0
 1: cs:Unconfigured
 2: cs:Unconfigured
 3: cs:Unconfigured
 4: cs:Unconfigured
```

The file contains very useful information, though of course it is not necessary to examine this file manually since Linuxha.net includes tools to do so.

The output above shows a single DRBD device in use, and various status indicators for that device. Most important are "cs" (Connection State) - which is set to "Connected", "st" (Status) - which is "Primary/Secondary" and "ld" (Local Disk) - set to "Consistent". These are the values that would appear under normal operating conditions indicating connectivity to the other node is present, the device is open and in use, and the data is consistent.

4.3 Scenarios Requiring Resynchronisation

DRBD attempts to minimise the amount of data that requires synchronisation. Consider each of the following scenarios:

- **Loss of Primary**
In this instance the application will fail-over and be made available from the remaining node. When the server later rejoins the cluster DRBD will just update (on the node joining the cluster), just those portions of disk that have been updated.

It might also be necessary to send updates from the failed node back to the live node, if such regions have not been updated since the crash on the node now running the application utilising the storage in question.

- **Loss of Secondary**
In this case any regions that are updated whilst the secondary is not attached to the cluster are replicated across when it rejoins the cluster.
- **Software Shut-down on Secondary**

The recovery requirements in this instance are the same as the loss of secondary - since the primary can determine the state of the secondary when it rejoins and just replicate any recent differences.

➤ Software Shut-down of Primary

In this case the secondary node is unable to determine the active change set that was taking place on the primary server. Hence when the application fails-over to the secondary, it forces that copy primary and when the original node comes back up and joins the cluster, must force that node "secondary". Hence in this state a full resynchronisation must take place.

The key point is that shutting down a node running an application incorrectly is **not recommended**. It is actually better to power that node off if shutting down the clustered application cleanly is not possible.

The amount of data that must be resynchronised is highly dependent not only on the amount of data that has changing but also on the number of extents DRBD uses for each active set. This is currently not configurable with Linuxha.net.

4.4 File System Support

Linuxha.net does not force the administrator to use a particular file system type to be used for the clustered storage. Of course a journalled file system is recommended since this allows speedier recovery. Hence it is recommended that that one of the following is used:

- ext3
- jfs
- reiserfs
- xfs

The software must also be able to allow the administrator to set options that might be applicable to their environment. For small test configurations "jfs" is the first choice since it allows very small logical volumes, (i.e. 20Mb), where as though "reiserfs" is potentially faster, it does not seem to work too well until the volumes are much larger – at least 50Mb in size.

Although it is possible to mix and match file systems throughout the applications being clustered for ease of use the recommendation is to choose just one particular file system and use it everywhere.

It should be noted that "ext3" does not offer on-line file system expansion and so from the availability aspect this is a definitive negative point, and for this reason the author does not recommend its use currently.

Of these file systems "xfs" differs from the other three since it is based on "extents" rather than blocks. Although the current versions of DRBD do indeed support "xfs" there do still appear to be some issues - so please ensure adequate testing is performed prior to using in a live cluster if this file system is to be used.

Thus the author recommends the use of "jfs" or "reiserfs". Both are stable, offer good recovery times and provide on-line file system expansion¹.

¹ It has been noted that RedHat Enterprise Linux 4 does not support either "jfs" or "reiserfs" officially, though packages are available they are unsupported.

5 Linuxha.net Solution Architecture

This section outlines the architecture used by Linuxha.net. The major components are defined and described, along with the files used to manage the cluster, applications are resources.

Remember that this document covers the features offered by version 0.9.0 of Linuxha.net, However it is unlikely that any major differences will invalidate this information for the 1.0.x releases.

5.1 Definitions of Terms

Before designing the software / cluster architecture it might be worth reiterating (and refining) a few terms that will be used throughout the remainder of this document:

- *Node* – A node is a machine that is part of the cluster. In the current scope of the software design a cluster must have two nodes, though both might not be available at any one time. The administrator might refer to a node by name, but the underlying architecture uses IP addresses defined in the cluster topology file in most cases, and does not rely on a single route for inter-node communication.
- *Application* – This is a discrete service that the cluster provides. Each application can have zero, one or more volume groups.

The cluster can consist of zero, one or more applications, (the maximum is not defined, but is determined only hard machine resources). Each application can have zero, one or more unique “application” IP address which will be present only whilst the application is running, and be the access point for clients to access the services offered by the application. Further IP addresses can be defined on different networks if desired.

- *Volume Group* – This is a collection of “logical volumes” – which are essentially block devices which contain file systems. A volume group is a collection of such block devices that are made available as a whole or not at all. Volume groups are provided by the “Logical Volume Manager”. Currently there are two versions supported under Linux - both versions of LVM are supported by Linuxha.net. Support for IBM's EVMS is to be added for post 1.0.0 releases.
- *Monitors* – These are programs that check the status of various components, either of an application (such as whether certain processes are running), or the cluster environment (such as network status monitoring).

Most monitors are run as modules of a sub-system calls “Lems” - the Linuxha.net Event Monitoring System. Each running application typically has a single “Lems” daemon running making use of several monitors. These monitors can fall into either of two categories – “system” or “application”.

The system monitors are standard monitors that are necessary to monitor the status of the clustered environment from the perspective of this particular application. System monitors typically monitor the IP reachability from the IP addresses for the application and importantly the status of file system data synchronisation. The “application” monitors are typically user-written programs or Perl modules that validate that the current application is still working effectively or not.

- *Service Addresses* – As stated above an application presents itself to clients via TCP/IP connectivity. Although some client/server architectures use broadcasts to ascertain hosts providing a service a more common approach is to connect via a defined IP address.

Each application may define zero or more IP addresses associated with it. It is actually possible to define multiple IP addresses either on the same network, or across multiple networks. Multiple IP addresses on the same network are typically used for applications such as Web servers, whilst providing addresses for multiple networks is typically used to provide “public” and “private” IP addresses (one for client access, the other for systems management).

5.2 Application Storage Management

Although it would have been possible to deal with simple block devices without resorting to volume manager support this was deemed unsuitable for the target environments. It is likely that any environment where high availability was taken seriously enough to invest in two servers would be making use of enterprise level features such as a logical volume manager (aka "LVM").

Hence it was decided that each application in the cluster will replicate based on the contents of separate volume groups. There are no limits to the number of volume groups a machine may have, and each volume group can have many file systems. The use of LVM means allows the administrator as well as the Linuxha.net software to validate and manage the environment more easily and is a core component of how replicated data is handled.

Since Linuxha.net product is built via the provision of duplicated synchronised data resources the software includes checks to ensure that each volume group that is configured as part of an application will be defined as identical on both nodes. Obviously if this is not the case the cluster is unlikely to work and the administration of the data sources will become overly complex. Such facilities ensure that once the volume groups have been defined on both nodes if necessary the software will be able to create and/or validate the volume group configuration for each application created.

As previously stated both LVM version 1 and LVM version 2 are fully supported. If there was a particular reason to make use of different versions on different hosts that is also supported – though obviously it would make administration more complex and hence it not recommended.

Some future release following 1.0.0 will include support for EVMS as well. Support for the Veritas Volume Manager is currently unlikely unless sponsorship of the license costs of the software can be met.

5.3 Managing Cluster Consistency Status Information

One of the most important aspects of the cluster administration is ensuring that information regarding data synchronisation is kept and managed very strictly. Early versions of the software attempted to handle much of this information explicitly. However with the use of DRBD as the replicated network block device, this facility is now mostly implicit - each DRBD device uses local meta-data to keep detailed state information regarding data consistency.

However some information must be retained for the cluster infrastructure as a whole, and hence the following directory will be created on both hosts to store such information:

```
/etc/cluster
```

This directory contains information on the application configurations, cluster topology, resource allocations, and current application state information.

Some of the more interesting directories stored here are:

```
/etc/cluster/.resources
```

This directory contains several sub-directories that indicate the resources that are in use on this particular node. Sub-directories such as "drbd", "ports" and "fsmap" can be found here (these are explained later in this reference).

```
/etc/cluster/application
```

Each application is allocated a separate directory in which the status, resources and definition of the application are kept.

Of course, since no storage is physically shared these directories are created on both nodes. In most cases the administrator only needs to enter configuration information on a single node -

the Linuxha.net software will copy any data that needs duplication as part of any cluster or application build process.

Apart from using the meta-data that exists for each DRBD device whenever an application is started on a node, the following file is created:

```
/etc/cluster/application/TIME
```

This contains the "UNIXTIME" (number of seconds since 00:00 01/01/1970) when this particular package last ran on this machine. This information is used when the contents of the status directory differs on each node. In this particular case the node which has the most recent time will be taken to have been run last and thus have the most up to date information.

5.3.1 Handling Specific Limitations

The approach taken for handling cluster failure scenarios works well when handling single points of failure - which is all most cluster software (commercial or otherwise) will aim to do. When one failure is followed by another before the first is resolved then manual intervention by the administrator will be required in many cases.

For example, if a node failed and then the remaining node failed without the first node being brought back in and any data synchronisation being completed then it will be the responsibility of the administrator to ensure that the applications are restarted on the 2nd node to ensure data synchronisation back to the first node can complete.

Note that failure of both nodes simultaneously, (i.e. if sharing the same power source and it fails), is not a problem unless data synchronisation has not been completed at that point (for example due to a previous error). In such cases, DRBD should recover data using the contents of the Meta-data for each replicated file system.

To help in such scenarios the cluster status reporting tool ("clstat") includes a report to indicate the times each application has been used on both nodes, so they can be started accordingly.

5.4 Cluster Status Daemon

It is important that cluster status configuration information is passed between both nodes if they running and able to communicate. Although the software does make use of "ssh", much of the time a more efficient method is to keep the information information available at all times and make it available via a daemon.

Hence although it is actually possible to force an application to start without such status daemons running, this approach should be used for emergencies only. Otherwise the start-up of an application will query a daemon to ascertain the current status of the application in question using such information to decide what steps should be taken.

When the two nodes start cluster status daemons around the same time they are said to be "forming" a cluster. If a cluster is already running, but on a single node, the remaining node is able to "join" the cluster at a later time.

The process of joining a cluster or forming it follows a set pattern of communication between the nodes to ascertain the status of each machine and to pass any current information between themselves that they deem relevant to the current status of the cluster. Once running in a steady state the daemons will be responsible for managing the status of the node – such as logging details, starting and stopping applications, saving state, and communicating with each other to check that they each are alive and functioning.

Since the status daemons communicate with messages that might actually cause application to stop or fail-over security is a key consideration. Hence the communication amongst the daemons will make use of Blowfish encrypted traffic – the password for the encryption will be kept as part of the cluster configuration with only root allowed to view such a file.

When the administrator starts the cluster service in "forming" mode the daemon will attempt to contact the other machine and if it can not be found it will occasionally probe for it up to a defined time limit. If it gets no response and has not been "forced" then it will abort the start-up of the cluster.

The actual mechanisms of the communication that the daemons use, both during the formation and during steady state operations can be found in the technical section of this publication - see section 5.

Finally it should be stated that although the cluster daemons are key to the functioning of the cluster the environment has been configured in such a way that if the cluster daemon fails (for example due to a software problem), it should be possible to re-start it without impacting application availability. Various failure scenarios (including loss of the cluster daemon) are dealt with in later sections of this document.

5.5 Cluster Lock Daemon

This daemon is responsible for handing out locks to various Linuxha.net processes. The locking is optional (i.e. the cluster will function if the lock daemon dies), but without it it is not possible to ensure serialised access to cluster resources.

These locks are only really necessary if multiple applications are running in the cluster, but it is recommended that it always runs, no matter the number of applications.

As with the main cluster status daemon the administrator is able to communicate and influence the functionality of the daemon via a simple command line utility.

5.6 Cluster Network Daemon

The cluster network daemon monitors the networking environment of a node. The work it undertakes is determined by the topology of the node on which it is running. The daemon will monitor the physical link status of all networks defined in the topology, typically 5 times a second, and on failure take appropriate steps.

Again, this daemon is optional. Stopping it, or not running it will not affect running applications, or even from new applications from starting - it simply means that network card failures will not be handled by the cluster.

When this daemon is running the loss of a physical connection on a card in the topology which supports physical link checking will result in either of the following actions taking place:

➤ IP Address Fail-over

If the specified network card is part of a network on this node that consists of more than a single network card then the IP addresses currently associated with this card are failed-over to another card.

When more than two cards are available in a defined network the card chosen is one that either has never been failed over to before, or was used the longest time ago.

This fail-over will affect all IP addresses associated with the card - both any IP addresses statically assigned or configured as part of any application currently running in this node.

➤ Application Fail-over

If a card failure is determined to be occurring too frequently for a given network² or a network only consists of a single network card, then it is possible that certain running applications will be migrated to the other node in the cluster.

Applications that are considered as suitable for this migration are defined as;

1. those which define IP addresses on this network.

² Too frequently is determined by the minimum time period between attempts to re-use the same card in the network.

2. those which are known to have a valid copy of data on the remaining node
 3. the other node in the application is up and running as part of the cluster
- No Action
- If the node is not running any applications or no applications rely on this interface and no alternative card is present or valid, then the failure will be logged, monitoring for this network stopped and no other action takes place.

If such cases it is the responsibility of the administrator the turn on monitoring for this interface at a later date once any problem has been found and resolved.

Details describing how the physical link status is checked can be found in later sections of this document.

5.7 Application Monitoring with “Lems”

When an application is started on a node a daemon known as “Lems” is started with the sole aim of monitoring the status of that application. Unlike some other cluster software products available Linuxha.net does not depend on a single cluster-wide daemon monitoring status of all applications, but instead uses one per (running) application.

The monitoring offered by the “Lems” daemon can be classified as belonging to one of the following categories:

- System
- Application

System monitors are those that are already provided as part of the base Linuxha.net product and should be running for every application to ensure that the cluster functions in the way expected.

However it is possible to turn off or even remove system monitors, depending on the individual requirements of the the application. The next section describes in outline each of the available monitors, indicating which can be considered as “optional”.

Application monitors are optional – it is possible to run an application without any application monitors, though usually at least one is made use of. It is very possible that a custom monitor might be required for a particular application and the interface to add such monitors is quite straightforward.

5.7.1 System Monitors

A section later in this document describes the “Lems” software in considerable detail, however in practise these monitors will perform the following roles (on a per-application basis):

- Checking and managing data synchronisation
- Checking IP services (if possible)
- Checking network link connectivity (if possible)
- Handling IP Address migration (when necessary)
- Flag Checking

Each monitor typically runs at different intervals or when triggered by another monitor – for example the “Ip Address migration” monitor only runs when the Link or IP address modules notice a problem.

5.7.2 Application Monitors

These are monitors that are optional and typically may differ on a per-application basis. Currently there is three such monitor already supplied with Linuxha.net– though typically only the first is required for most environments:

➤ Process Monitor

This monitor is used to check to see whether a certain process is running, and if not run a command to restart it. A certain number of restarts are allowed in a set period of time after which typically a fail-over of the package to the other node is undertaken.

The monitor provided is driven by a configuration file that the user must create and obviously entirely depends on the application being monitored.

From experience it has been found that the abilities of the standard process monitor should meet most requirements for simple application process monitoring and restarting. One limitation it does have is that it does not handle any dependencies between the application components – for example if you have a package which consists of a database and an application, you may wish to restart the the application if the database fails, but if the

application fails the database does not need to be restarted. In such scenarios the author of the process monitor configuration file must take care of the dependencies – the software is currently unaware of them.

- File System Capacity Monitor
This monitor will check the available free disk space in one or more file systems and when it drops below a certain threshold it will return a known error code. This is useful for failing over applications if they rely on certain local free resources, (such as space in “/tmp”).
- Swap Space Free Monitor
This checks the available disk space and if it drops below the specified threshold it will again respond with an error code. This can be used to fail-over or shut-down applications to ensure other applications running on the same server continue to function without risking out-of-memory conditions.

Since the “Lems” daemon for a running application is quite critical the main cluster daemon actually scans for a running daemon when an application is live, and if not found will start one.

5.8 Cluster Utilities

Often people have the perception that clusters are complex, and although in some instances this is undoubtedly true, the aim of Linuxha.net is to make cluster configuration and management as simple as possible. Experience has shown that to make a cluster manageable it must be straightforward to build initially, change later and report on whilst running. Further since this is high availability software as much work as possible should be able to be performed without downtime required for either the applications in question, or worst case, the complete cluster software.

As of version 1.0.0 a complete set of tools and utilities are provided allowing the cluster to be built and configured, the applications built and cluster status to be ascertained. All daemons provide logging when required, and the utilities are written to be usable in various cluster operating conditions.

For example the tools to “build” the cluster definition can be used both initially and when the cluster is actually running. In the latter case certain changes are communicated to the cluster daemons which are dynamically updated.

For the utilities to work the nodes must support “Ssh” services without recourse to a password prompt. Indeed “Ssh” configuration between the nodes is a key requirement, and information on how to configure “Ssh” can be found starting on page .

5.9 Third Party Software

As with most software installations Linuxha.net has some dependencies. However the majority of these dependencies are very common, and are likely to be already in place in most environments, whilst the less common components (certain Perl modules) are bundled with the core product and will be compiled and installed automatically if required.

The largest dependencies **not** covered by the standard package are;

- Perl
- Kernel Header Files
- Gcc
- Parser tools (Bison/Flex)

Most modern environments have these installed by default. Without them the installation will fail, (the reasons will be found in the post installation log file).

Part II:
**Software Installation and
Build of a Sample Cluster**

6 Installing Linuxha.net

6.1 Hardware requirements

Prior to installing the necessary software there are some basic points that should be considered before a cluster can actually be built. These considerations hold true whether building a virtual machine environment, (for example using "VmWare"), or a cluster consisting of two different physical machines.

- **IP Connectivity**
The environment must be configured to ensure that at least one, (ideally two or more) network interfaces are available on each server. Each should be able to communicate with a similar interface on the other machine. In theory it is possible to have different network topologies on both servers, but in practise this is not recommended since it increases complexity.
- **Network Throughput**
Given that the basis of the clustered storage is provided by an IP-based replication scheme it is important that the performance offered by the network used for this traffic exceeds the expected throughput. If this is not the case this network link can become a bottleneck. For larger applications cheaper Gigabit Ethernet cards are recommended – 32bit versions to fit in cheaper hardware are now less than £30 (about \$50 US) – and offer at least 3 times the throughput of 100Mbit/cards.

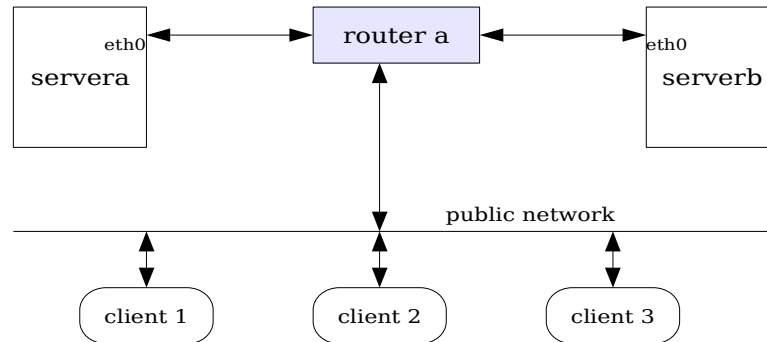
6.2 Typical Hardware Configurations

Through out this document various configurations will be used to describe the software functionality. This section aims to provide some variations on the types of hardware configurations that the administrator might consider - all of which should be determined by the requirements of the application to host, not the Linuxha.net software itself.

It should be noted that this section gives only some of the possible permutations. If the reader has any questions regarding the suitability of a configuration then please contact the author using the email address given at the start of this document.

6.2.1 Configuration 1 - No redundancy

This is the simplest configuration and is although offers redundant servers, is prone to several problems as described below.



Problems:

- (1) No network redundancy - loss of a single network card will result in a complete fail-over of any applications on the affected node.
- (2) Risk of excessive "network partitioning". Network partitioning should be avoided at all costs. This term refers to the situation when the two nodes are unable to communicate - the software then has to decide whether the other node has failed, or the network only. With only a single network connection, even if using multiple cards, it is not possible to distinguish which has occurred.
- (3) Sharing the data synchronisation traffic with client-server traffic is not recommended, since one might impede the performance of the other.

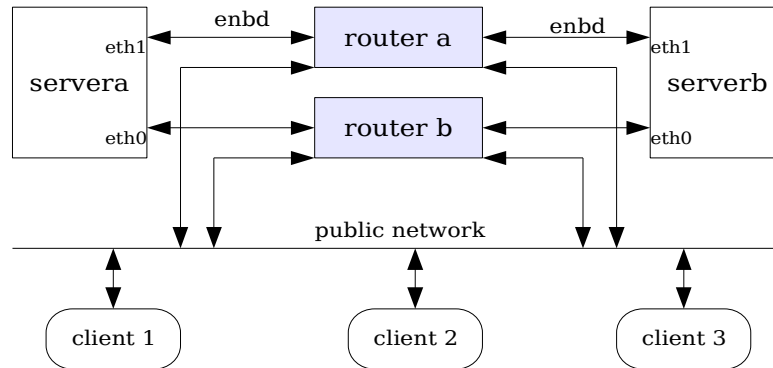
Point 2 above is a risk, since if the network condition is resolved the administrator might be faced with two servers attempting to run an application / using an IP address, which is obviously disastrous. The Linuxha.net has an inbuilt ability to recognise this situation if both nodes can communicate at a later date, but any occurrence must be considered to be very bad.

Note that "handling" the problem of both nodes running the same package simultaneously is handled by killing off a node immediately. This is not a suitable but an immediate software reset.

For the above reasons using this cluster topology is strongly discouraged. Indeed given the cost of two servers, an additional network card each to define an alternative communication channel should not be beyond the budget of any hardware solution.

6.2.2 Configuration 2 - Basic network redundancy

The least level of network redundancy is offered by the following type of solution.

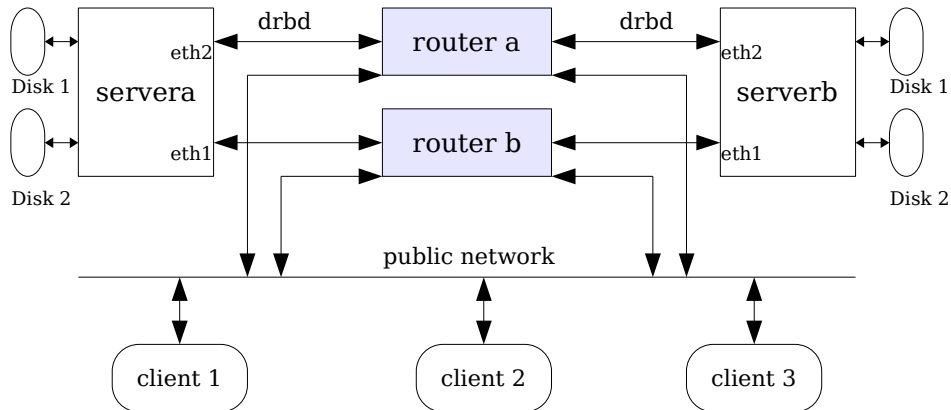


In this configuration the disk synchronisation traffic stills has no redundancy, but at least the nodes are able to communicate across two networks, meaning that loss of a single card, or even router, will not result in a “partitioned network” scenario - even if it does mean that one or more applications may need to fail-over.

In such configurations loss of “eth1” will lose the data synchronisation path, leading to stale remote copies, but the application will not fail. Loss of “eth0” would result in all client traffic being unable to communicate (each network must use unique cards and reside on different subnets), thus requiring any applications presenting an IP address on this network to be failed over to the other node.

6.2.3 Configuration 3 - Local Storage Redundancy

In this configuration not only are there multiple network paths, but also mirrored storage on each server. This storage is typically via hardware or software RAID. Either the traditional “md” device driver or the newer “device mapper” should work to provide software RAID for Linuxha.net without issue.

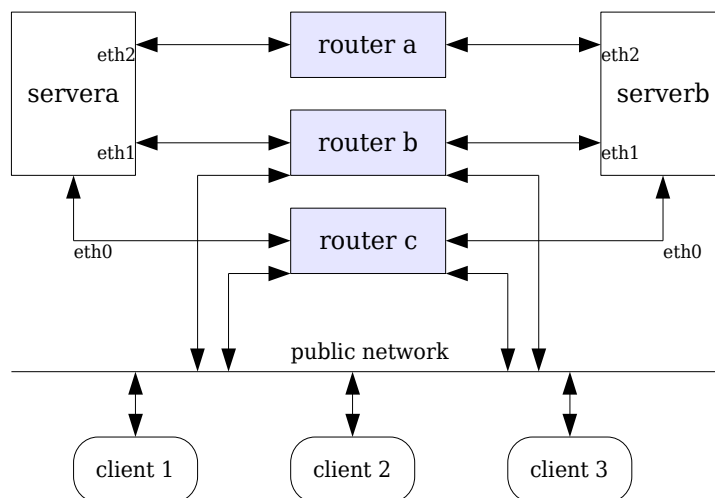


Obviously in such a scenario the solution should be more redundant again to hardware failures. Minimising the number of types a “hardware-failure” scenario occurs reduces the number of instances were a full data resynchronisation will be necessary.

However the above application still only offers “eth1” for application IP traffic, meaning failure of a single network card could result in application fail-over being required.

6.2.4 Configuration 4 - Complete network redundancy (Multi-pathing)

The following diagram show this network topology. Of course it could be combined with the hardware or software RAID solution just shown.



In this configuration the following roles are given to each interface:

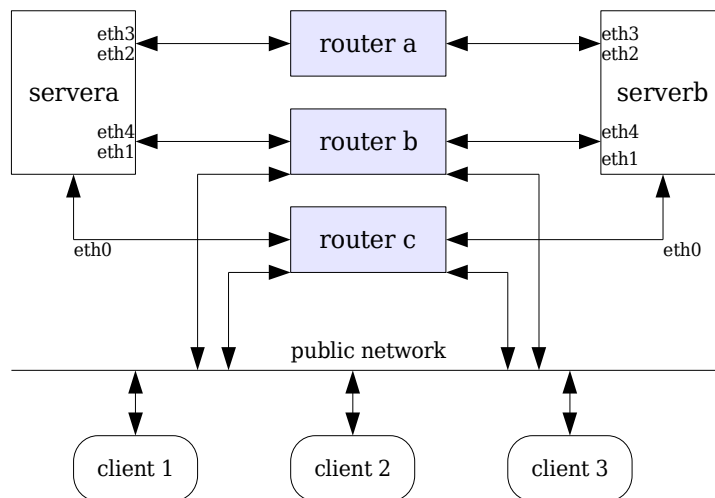
- eth0 Connected to the “public” network only - by default clients will communicate via the IP addresses associated with this node.
- eth1 This is part of the same “public” network but under normal circumstances does not have any IP addresses associated). If “eth0” fails it will assume all the IP addresses for that interface on the node in question - hopefully without disruption to existing client access.
- eth2 Dedicated to DRBD traffic - in fact could be a simple point-to-point link (via cross-over cable), rather than a routed network.

The following important points regarding this configuration should be noted:

- (1) By only providing a single network card in the network used for “DRBD” traffic loss of a card (on either host) will result in stale remote data copies. Despite this such a topology is commonly used.
- (2) By providing “eth1” as a fail-over card for the public network, the loss of “eth0” will not require any applications to fail-over to the alternative node. Instead the cluster network daemon will make the IP addresses quickly available on “eth1” rather than “eth0”.

6.2.5 Configuration 5 - Multiple Public Networks

In this configuration applications may define IP addresses that exist on two different networks (for example for local and remote access in a cross-campus configuration).



In the above configuration note the following points:

- (1) Two interfaces have been defined for each of the public networks, two networks for public IP addresses, though in the above example only a single card “eth0” has been allocated for DRBD network traffic - an additional card could add redundancy.
- (2) 5 network cards is a configuration that most lower-end servers will not be able to accommodate (lack of physical PCI slots).
- (3) The network monitoring software does not differentiate between any network - if it is defined in the cluster topology then it will be monitored. Of course the administrator can manually turn off and on monitoring of individual networks for each host from the command line if necessary.

Older versions of Linuxha.net supported bonding configurations explicitly. Currently that is not the case (since IP fail-over is much improved since earlier releases.) Bonding support has not yet been tested either for public or private network connections and so is **not** a currently supported option.

6.2.6 Specific Hardware Concerns

It has recently come to light that certain hardware RAID controllers can suffer from very poor performance when using DRBD. At the moment this appears related to specified controllers manufactured by "3ware".

Hence if the hardware for a production cluster is not already in place it is always worth querying the DRBD mailing list (after scanning previous entries) to ascertain whether the hardware in question will indeed be suitable.

Details of gaining access to the mailing list ("drbd-user") can be found at the following URL:

<http://lists.linbit.com>

6.2.7 Hardware Configuration Conclusions

Although the number of supported configurations is substantial, the following points should be considered when deciding any solution:

- Locally mirrored storage is highly desirable. Whether this is RAID 1 or RAID 5 does not matter - just the redundancy offered by not losing a server from a single disk failure. Further it does not matter whether this is software or hardware RAID.
- Servers should be in different locations - that is an ideal - but one of the key advantages of offering a replicated storage solution over TCP/IP is that the servers can be any distance apart, as long as a suitable level of bandwidth for the solution is available.
- Servers should be on different power supplies, or at least UPS protected. Otherwise both servers are at risk to a shared single point of failure. Use of DRBD means that a full resynchronisation would not be necessary for recovery, but still not a desirable situation to be in.
- DRBD does not support built-in diverse routing of replication data. However since the network cluster daemon can be used for IP fail-over for the replication network, an alternative route (and routes!) should be seriously considered for production configurations.
- Multiple interfaces for client access should really be considered mandatory.
- Separation of DRBD replication data and client/server information transfer is recommended if you have a high enough number of interfaces. Alternatively ensure Gigabit interfaces are used throughout the solution.
- Bonding of interfaces is not currently supported - though support may be introduced at some point in the future.
- If possible scan the "drbd-user" mailing list before purchasing any hardware - especially since it has been shown certain RAID controllers can suffer from very poor performance when using DRBD 0.7.8 and although Linuxha.net now makes use of 0.7.10 some uncertainty still remains regarding this point.

6.3 Environment Configuration

If an existing version of Linuxha.net is currently installed on the cluster then please pay attention to the information given on page .

For the cluster software to work both machines in the cluster must have compatible versions of the Secure Shell installed and the Secure Shell Daemon must be running, or be made available for the "Internet Super Server" (either "inetd" or "xinetd" depending on your distribution).

The Secure shell must be configured to ensure that root equivalence to both the remote machine and itself are available on all IP addresses defined for both machines. Further this equivalence must not require a pass phase to be entered.

For information on how to configure SSH to function in this manner, please see "Appendix C: Setting up SSH", starting on page .

The Linuxha.net package when installed requires that the following software is also available:

- Perl
- Bison
- Flex
- Gcc (and suitable tool chain)

If any of the above are not installed the post installation will fail and the software will not be usable. In such cases re-installation of the software can be attempted once the missing software has been installed.

More details on the exact software requirements can be found in the next section.

Prior to version 1.0.0 the software is available only in the "Tarp" package format. How such packages can be installed is discussed later in this section. For version 1.0.0 and above the following package formats should be available:

- *Tarp* - generic package that can be used on any Linux distribution with minimum dependencies.
- *RPM* - native package format that can be used for RPM-based distributions, such as Fedora, RHEL and Mandrake.
- *Autopackage* - generic package format designed to supply the native package format in use on any particular distribution.

6.4 Kernel Configuration

Linuxha.net software has been designed to function on all modern Linux kernels - from 2.4.10 through to 2.6.11.

Currently the version of DRBD used for Linuxha.net is known to be compatible with the very latest kernels (up to and including 2.6.11).

The "LVM" functionality must either be installed as a part of the kernel or installed as a module when the server boots. If you are using a distribution such as "Slackware" the following entry in "/etc/rc.d/rc.local" would suffice:

```
/sbin/modprobe lvm-mod
```

"Linuxha" supports both versions of LVM that are available for Linux (version 1 and 2). Support for EVMS is not currently available, but is planned following 1.0.0.

The Linuxha.net software probes for the current support of LVM when an attempt to build a cluster is made - and will produce a suitable error if support is not available. Most modern distributions require no alteration or special steps to ensure such support is available.

If software RAID support is required to locally mirror storage then support for the required RAID device driver is required (typically known as "md" - though the Linux 2.6 device mapper may also function in this manner).

6.5 Prerequisite Software

Almost all of the code that comprises of the main Linuxha.net package is written in Perl. Thus the first requirement is to have a version of Perl which has been tested as compatible. Currently it is recommended that either of the following versions be used, (which are those for which it has currently been tested):

- 5.6.1 or above
- 5.8.0 or above

Testing on the vanilla version of 5.8.0 has revealed that setting "\$0" does not change the text associated with the process name and arguments - this is a requirement of Linuxha.net. Although a work-around is now built in later versions of 5.8 are recommended.

Note that some distributions supply patched versions - for example Redhat Enterprise 3 supplies version 5.8.0, but does not suffer from the above problem.

Please note that the current release of Linuxha.net does not make use of "ithreads" - the current implementation of threads in Perl was found not to improve efficiency.

The Linuxha.net software is available in the generic "Tarp" format, an RPM package or an "Autopackage" package, with other formats being considered for future versions. If the administrator wishes to make use of "Tarp" packages follow the instructions in the next section.

6.5.1 Installing Tarp Package Management Software

To make use of such packages the administrator must first install "Tarp". For a recent download of the tool set, and Installation instructions please see the project's home page:

<http://www.linuxha.net/index.pl?ARGS=findproject:tarp>

Please note that this software requires the use of the Korn shell or "zsh" running as korn. If the target machines do not currently have a shell called:

```
/usr/bin/ksh
```

then check to see if either "/bin/ksh" or "/bin/zsh" exists, and if so create a link - for example if "/bin/ksh" is available:

```
# ln -s /bin/ksh /usr/bin/ksh
```

Since the standard "KornShell" (not the earlier "ksh") is now available under an open license this is now the recommended software to use. Information and downloads of this package (if not supplied by your vendor) can be accessed here:

<http://www.kornshell.com>

Once the shell has been set up to install the Tarp package, perform the following steps – in this instance using version 1.3.3 which has been downloaded with the following full pathname:

```
/tmp/tarp,1-3-3.tarp.gz
```

Firstly make a sub-directory and change to it:

```
# mkdir /tmp/install
# cd /tmp/install
```

Now extract the files from the archive available in the parent directory:

```
# tar xvzf ../tarp,1-3-3.tarp.gz
```

Now the local copy of the "tpinstall" command can actually be used to install the "Tarp" package itself. This is done using the following command:

```
# usr/local/bin/tpinstall -d $PWD/.. -i -p tarp -v
```

This should produce output similar to the following:

```
MSG : Package database directory: /var/adm/tarpdb
MSG : Installing from file tarp,1-3-3.tarp.gz
MSG : Using temporary directory /var/adm/tarpdb/tpinstall-682
MSG : No dependencies required for package.
MSG : Number of dependencies to install: 0
MSG : Beginning package extract to /var/adm/tarpdb/tpinstall-682...
MSG : Package extraction completed successfully.
MSG : Saving files that will be overwritten...
MSG : No files required saving.
MSG : Installing package files...
MSG : Files installed: 14
MSG : Setting Correct package permissions + ownerships...
MSG : Removing spooled package files...
MSG : Package tarp installed successfully.
MSG : Package tarp successfully committed.
MSG : Cleaning up directory /var/adm/tarpdb/tpinstall-682
```

Once this step has been performed on both machines, please ensure that the PATH variable includes the directory "/usr/local/bin" directory – this should be added to the root user's profile if necessary before continuing.

To check that the software has been successfully installed run the following command:

```
# tplist
```

This should produce output similar to:

Package	Version	Status	Description
tarp	1.3.3	Committed	Simplified package manager

The manual pages for the packaging programs are installed into the following directory:

```
/usr/share/man/man1m
```

Some Linux distributions do not include the “m” section by default, and if this is the case please update the following file:

```
/usr/share/misc/man.conf
```

Ensure that the directive “MANSECT” is updated to include the “m” section, for example:

```
MANSECT      1:8:2:3:4:5:6:7:9:tbl:n:l:m:p:o
```

Finally the temporary directory used for the installation of the “Tarp” tools can be removed:

```
# cd /tmp
# rm -rf install
```

6.6 Installation of the “linuxha” package

If a previous version of the package is already installed and in use then the first step is to stop any packages and any cluster daemons. Upgrading a running cluster is usually supported, see later sections in this document if that approach is more suitable.

To remove a previous installation first use the following commands:

RPM based installations:

```
# rpm -e linuxha
```

Tarp based installations:

```
# tpremove -v -p linuxha
```

Autopackage based installations:

```
package remove linuxha
```

Now the latest version of the package should be installed, using the command appropriate for the package type to install. All commands assume that the package is available in the local directory.

RPM based installations:

```
# rpm -i linuxha-1.0.0-1.i586.rpm
```

Tarp based installations:

```
# tpininstall -i -p linuxha -v
```

Autopackage based installations:

```
bash ./linuxha-1.0.0.x86.package
```

The RPM installation is silent, whilst the Tarp package install will show something similar to the following:

```
MSG : Package database directory: /var/adm/tarpdb
MSG : Installing from file linuxha,0-9-2.tarp.gz
MSG : Checking for /var/adm/tarpdb/linuxha
MSG : Using temporary directory /var/adm/tarpdb/tpinstall-11940
MSG : No dependencies required for package.
MSG : Number of dependencies to install: 0
MSG : Beginning package extract to /var/adm/tarpdb/tpinstall-11940...
MSG : Package extraction completed successfully.
MSG : Running pre-install script...
MSG : Pre-install script completed successfully.
MSG : Saving files that will be overwritten...
MSG : Files Saved: 310 (3844 Kb)
MSG : Installing package files...
```

```

MSG : Files installed: 310
MSG : Setting Correct package permissions + ownerships...
MSG : Running post-install script...
MSG : Post-install script completed successfully.
MSG : Removing spooled package files...
MSG : Package linuxha installed successfully.
MSG : Pre installation script output can be found in /tmp/tpinstall-pre-linuxha-11940.stdout
MSG : Pre installation script errors can be found in /tmp/tpinstall-pre-linuxha-11940.stderr
MSG : Post installation script output can be found in /tmp/tpinstall-post-linuxha-11940.stdout
MSG : Package linuxha successfully committed.
MSG : No TMP_BUNDLE_DEPOT defined (no tmp depot to remove)
MSG : Cleaning up directory /var/adm/tarpdb/tpinstall-11940

```

If a Standard Error file is mentioned in the tarp then please ensure the contents are checked for potential failures. Any failure should be investigated since ignoring post-installation failures may cause the product not to function correctly. For the Autopackage and RPM installations pre and post installation script output can be found in obviously named files in root's home directory.

If Autopackage or RPM indicate potential errors please review the output generated to check for errors. If either case standard output and standard error logs will be found in root's home directory.

Once complete use the following commands to ensure that the package has been installed successfully:

Tarp based installations:

```
# tplist
```

This should now output the following:

Package	Version	Status	Description
linuxha	1.0.0	Committed	Linux Replicated HA
tarp	1.3.1	Committed	Simplified package manager

RPM based installations:

```
# rpm -q linuxha
```

This should simply output the package name of the currently installed version:

```
linuxha-0.9.2-1
```

Autopackage based installations:

stuff here

Following successful installation of this package everything should now be in place ready to start the cluster configuration and build. The process of configuring the cluster is straightforward - **assuming time is taken to read either this reference or an appropriate cluster configuration guide document.**

6.6.1 Manual Check for LVM Support

Although the build of the cluster will check for the availability of LVM on both nodes it is often easier to resolve such problems at the earliest possible stage. Hence the administrator is recommended to run the following command on both nodes at this point:

```
# /sbin/cluster/utlils/lvmtool -lvmttype
```

This command will output either "1" or "2". If no output is given LVM is not configured or installed correctly. Please review the packages / software available for the distribution in question and resolve before continuing.

Earlier versions of Linuxha.net required that a patched version of the LVM user-space tools be installed to ensure compatibility with the network block device. Now that Linuxha.net uses DRBD rather than ENBD this is no longer the case.

6.6.2 Update the Superuser's PATH

All of the Linuxha.net commands are installed in a non-standard directory, and so it is recommended that the following directory is added to the default PATH for "root":

```
/sbin/cluster
```

Lines such as the following could be added to either ".profile", ".bashrc", ".kshrc" or ".zshrc", depending on the distribution and choice of shell. Since software components are either stand-alone scripts or Perl programs and so the choice of shell used for the "root" account does not affect their functionality.

Typical entries to add to the ".profile" might be:

```
export PATH=/sbin/cluster:$PATH
export MANPATH=/usr/local/cluster/man:$MANPATH
```

6.7 Installation differences between 2.4 and 2.6 kernels

Early versions of Linuxha.net that made use of ENBD as the network block device required some additional installation steps. Since the migration to DRBD **this is no longer the case**.

With the current versions of the software the installation and use of the software should be identical on both 2.4 and 2.6 based kernels.

6.8 Upgrading from previous Installations

This section covers the "basic upgrade" scenario. This approach is safest in all configurations. A separate section covers possible on-line upgrades, though this is only recommended for those with considerable experience of the product. Although this basic process is straightforward; there are still several points to bare in mind when upgrading to a newer version of Linuxha.net.

In all cases please review the documentation section of the web site. It may contain a guide describing the upgrade and migration process from to to certain release levels. Failure to read the appropriate guides may result in data loss.

- Installation will require that the cluster is not running. If the environment in question is installed and managed using any of the packaged solutions (Tarp, RPM or Autopackage) the installation will not be allowed if the cluster is still running. However if installation directly from the tarball takes place you will manually need to stop the cluster first, for example by running:

```
# clhalt --force
```

- Simply installing over the top of a previous version **is now possible**. However it is only recommended that this approach be taken if the current version installed is 0.9.0 or greater.
- Unless specified in any release-specific notes installation **will not** require the rebuild of either the cluster configuration or any currently defined applications. Of course the packages might need to be rebuilt to take advantage of additional facilities newer versions might offer.

7 Building the Cluster Configuration

The aim of this section is to discuss how it is possible to take information on the hardware configuration of the two machines that are to make up the cluster, and generate a configuration file to define the cluster based on this information.

Since this is a reference it is recommended that the appropriate “Cluster Configuration Guide” be referred to as well until the administrator is familiar with the software.

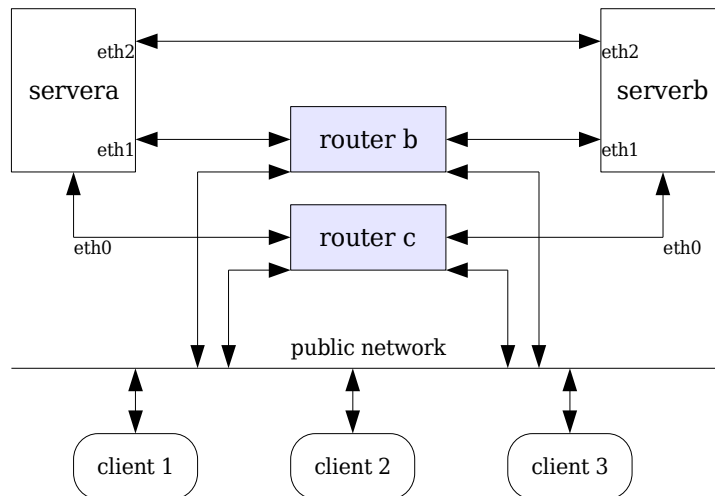
Typically most Linux software relies on the administrator “hand-cranking” the solution. Due to the complexity and potential for mistakes it was deemed early in the design that as much as possible configuration of the cluster would be automated, hence the actual configuration necessary can be performed very quickly.

It is not improbable for an administrator of the software to install, build and configure a cluster including a standard application, (such as Apache) in under two hours (excluding data synchronisation).

7.1 Detailed Example Cluster Configuration

A common cluster configuration is now shown as a sample configuration. Different examples of possible server configuration, and any important considerations for such approaches is shown previously (see information starting on page 42).

The information below shows the configuration of the sample environment used for the remainder of this section:



In the above diagram each machine has three interfaces, one (“eth2”) dedicated for the DRBD data replication traffic (and configured using a cross-over cable), whilst the other two (“eth0” and “eth1”) being connected to a “public” network – the network that will be used by clients when accessing the application.

The lack of redundancy of the “drbd” is a slight concern, but typically still a common scenario. For a production environment the network infrastructure should probably be monitored anyway and hence suitable recovery actions started as soon as possible.

Both “ServerA” and “ServerB” will use a volume group “app01vg” which will provide the storage for the application to serve – in this example it will be an Apache web server.

The table below shows some basic network topology details for the sample cluster:

ServerA		ServerB	
Static IP (on Eth0)	172.16.177.1	Static IP (on Eth0)	172.16.177.2
Static IP (on Eth2)	192.100.100.1	Static IP (on Eth2)	192.100.100.2
Network "drbd"	eth2	Network "drbd"	eth2
Network "public"	Eth0 / Eth1	Network "public"	Eth0 / Eth1
DRBD network	drbd	DRBD network	drbd

Notice the following important points regarding server configuration:

- (1) Any "standby" interface on a server **must not** be allocated an IP address - though it must be active - that is be visible via "ifconfig -a".
- (2) All networks configured must be on different subnets otherwise IP routing to the host will not work in the way the administrator might expect.
- (3) Heartbeat traffic is not defined as being used on particular interfaces or networks. All are tried and when one is found that works it is relied on until it fails.
- (4) The actual cluster configuration consists of many more entries - the above are just the minimum that must be defined to use the sample cluster topology.

Now that basic cluster topology has been defined, below are the basic settings for the actual sample "Apache" application. Again these are the **basic** details, many optional parameters are possible - these are described in detail from page onwards).

Property	Value
Basic Application Details	
Application Name	apache
Application Description	Provide access to Apache v2 web server
Application network / IP Address	"public" / 172.16.177.200
Process Monitor (#1) Details	
Monitor description	Web Server Processes
Name / Pattern to monitor	httpd
Process Owner	nobody
Minimum number running	1
Maximum number running	10

Property	Value
Volume Group (#1) Details	
Volume Group Name	app01vg
Logical volume 1 – name	admin
Logical volume 1 – mount point	/apache/admin
Logical volume 1 – size	20 Mb
Logical volume 2 – name	docs
Logical volume 2 – mount point	/apache/docs
Logical volume 2 – size	20 Mb
Logical volume 3 – name	logs
Logical volume 3 – mount point	/apache/logs
Logical volume 3 – size	20 Mb

As you can see the configuration is very modest – in fact this is an implementation built using two virtual servers in “Vmware” – though User Mode Linux might be just as effective or even Xen.

If you wish to build a similar sample package you can find the sample configuration available as a package using the following URL:

<http://linuxha.net/index.pl?ARGS=findproject:linuxha-apache>

The above logical volume requirements do not include meta-data volumes that will be created by Linuxha.net for each file system. Given that each meta-data volume must be 128Mb this can lead to a significant overhead when using multiple file systems though it is absolutely required to enable correct and accurate handling of data replication on a per file system basis.

7.2 Initial Cluster Build

When performing the initial cluster build the applications that are to be clustered do not need to be available, though if they are currently mounted and in use this should be on the machine that is considered the “primary” machine – the machine that alphabetically would come first if you compared the two host names. Hence if “ServerA” showed the following it would not present any problems at this stage.

Filesystem	1k-blocks	Used	Available	Use%	Mounted on
/dev/sdal	1542156	1037104	505052	68%	/
/dev/app01vg/admin	19248	352	18896	2%	/apache/admin
/dev/app01vg/docs	19248	1180	18068	7%	/apache/docs
/dev/app01vg/logs	19248	168	19080	1%	/apache/logs

Using the information shown previously we need to create a configuration file defines the cluster. To build the cluster the following file needs to be created:

```
/etc/cluster/clconf.xml
```

An example configuration file that can be used as a template can be found in this directory - this is part of the standard Linuxha.net product distribution.

This configuration file describes the cluster, and like all configuration files that are meant to be edited by the cluster administrator, is a simple XML file. Using the information shown previously it could be defined as follows. Notice that some of the information is taken from the table shown on page , but other elements shown have yet to be defined. Please see the table following this example configuration for summary details:

```
<?xml version="1.0"?>
<clconf>
<global>
    <name>cluster1</name>
    <version>0.3</version>
    <data>replicated</data>
    <datadetail>drbd</datadetail>
    <dataprotocol>C</dataprotocol>
    <logdir>/var/log/cluster</logdir>
    <key>mykey</key>
    <port>9900</port>
    <clport>9849</clport>
    <clnetdport>9850</clnetdport>
    <portpool>9901,9999</portpool>
    <maxblockdevs>50</maxblockdevs>
    <drbd_network>drbd</drbd_network>
    <echotype>ICMP</echotype>
</global>
<timings>
    <keepalive>3</keepalive>
    <warn>7</warn>
    <dead>12</dead>
    <clusterform>300</clusterform>
</timings>
<node>
    <name>servera</name>
    <network name="public" cards="eth0,eth1"/>
    <network name="drbd" cards="eth2"/>
</node>
<node>
    <name>serverb</name>
    <network name="public" cards="eth0,eth1"/>
    <network name="drbd" cards="eth2"/>
</node>
</clconf>
```

7.2.1 Field-by-Field explanation of "clconf.xml"

If the sample configuration file is used as a template the creation of this file just takes a few minutes. A more complex cluster configuration obviously will take more time. Each of the fields in the file should be understood before continuing, starting firstly with a description of the "Global" settings:

Field	Purpose
Global.name	The name of the cluster – this can be any length, but should be alphanumeric and not contain any spaces.
Global.version	This defines the version of the configuration file format that is being used – at the moment this should always be "0.3". This provides a sanity check to ensure the software and configuration file are compatible.
Global.data	This will always be "replicated" for the moment, though "shared" might be supported in the future when shared storage clusters are supported.
Global.datadetail	Current versions of the software only support "drbd" - older versions supported "enbd" and "nbd" but such values are no longer valid.
Global.dataprotocol	An optional setting that defines the DRBD protocol to use for data replication. If not present the safest setting of "C" is used. Can be set to "A", "B" or "C" - view DRBD documentation for information regarding the differences each protocol makes. Note that "C" is the safest (and slowest) protocol supported.
Global.Logdir	The name where log files are written to by the main utilities. (This is not really used properly, currently changing this from "/var/log/cluster" is not supported).
Global.key	This is critical. It defines a key which is used to validate all communication between the nodes in the cluster. This should be an alphanumeric string and should include no spaces or periods. This information is then used as part of a random key that encrypts every client/server communication between cluster nodes and services. Hence the cluster configuration file must be readable only by "root".
Global.port	Indicates the port number on which the cluster daemon should listen on, and send requests to on the remote node. This port can be any you wish, though typically 9000 is used unless the administrator has a good reason to use another.
Global.portpool	A min,max list of ports that define a range of ports that should be scanned when looking to allocate a port for each replicated file system. This same range of ports must be available on both nodes, and a range covering at least 50 ports should be specified. The number available determines the total number of file systems that the cluster can manage – so 50 is more than enough for most cluster topologies.
Global.maxblockdevs	The maximum number of network block devices that can be defined in the cluster. This currently must provide the same number of devices as the range given in the "port" pool, since each network block device requires a network port and a block device.
Global.clport	The network port allocated on both nodes of the cluster that can be

Field	Purpose
	used for the cluster lock daemon. This is optional, but if not present cluster lock support will be disabled - which is strongly discouraged.
<code>Global.clnetdport</code>	The network port allocated to both nodes for the cluster network daemon. Again this is optional, though if removed it will mean that network fail-over functionality for the cluster will be disabled.
<code>Global.drbd_network</code>	The logical network name defined in the "networks" section which is used for communication of DRBD data replication traffic.
<code>Global.echotype</code>	<p>The type of request sent between the nodes when ascertaining whether a network is suitable for heartbeat testing. Heartbeats use a dedicated protocol, but use this echo request type when probing a network for the first time.</p> <p>The default value is "ICMP", but the cluster can use "UDP" or "TCP" - though these will likely require the administrator to configure such services explicitly on each node.</p>

The next table defines the values given in the “timings” section which are critical for deciding how long to wait before a node is considered as being down, how long to wait to form a single node cluster, and how often heartbeat packets should be broadcast.

All times are given in seconds, and can include fractions if deemed necessary or useful.

Field	Purpose
<code>Timings.keepalive</code>	How often signals should be sent between the cluster daemons to ensure that each node is considered “alive”. A typical range of values is between 1 and 10 seconds, though a value lower than 3 is not recommended for all existing software versions.
<code>Timings.warn</code>	<p>The duration of time after which a warning is logged indicating that no heartbeat packets from the alternative node has been made available. Typically defined as twice the “keepalive” value.</p> <p>Although this value does not alter any characteristics of the cluster it is very useful since it will cause a warning to be given after failing to contact the alternative node at this time. This can then be used to alter the timings if the nodes are particularly busy.</p>
<code>Timings.dead</code>	After this period of time the node that did not respond is marked as “dead” (assuming it is not decided that network partitioning has occurred). In this instance the daemon start start to take over the affected applications. Typically defined as between 5 and 30 seconds.
<code>Timings.clusterform</code>	This defines the default length of time one cluster daemon will wait for the other daemon to respond when attempting to form a new cluster. Typically this is set between 60 and 300 seconds. It can be overridden on the command line if so required.

The final section of the configuration file defines which nodes will form part of the cluster, and also defines the network topology that should be considered as being part of the cluster. If an interface is not defined in this list then it will not be monitored by the cluster network daemon, for example.

Field	Purpose
<code>Node.name</code>	The name of a node – must resolve to an IP address that both nodes can use – though does not matter which of the various interfaces that are available to the nodes.
<code>Node.Network/name</code>	A single word which is used to label one or more network cards that can be used to host a series of IP addresses.
<code>Node.Network/cards</code>	A comma-separated list of network interface names that form this network. The points following this table should be considered when defining the network topology for each node.

When defining the cluster networking configuration, consider the following points:

- The cluster must include at least one defined network - this being used for the DRBD replication traffic, as used in the "drbd_network" element in the globals section.
- The list of cards given for a network must currently be Ethernet cards, using the naming scheme of "ethN".
- The minimum number of cards defined for a network is one, whilst the maximum is not defined.
- Both nodes in the cluster must provide the same logical networks, though the interface names, and numbers can be different on each node.
- A particular interface card can only be part of a single "network" entry.

As stated, the globals for "portpool" and "maxblockdevs" are closely related - the lowest number of these two settings define the maximum number of replicated file systems that the cluster can deal with.

Much more information on resource allocation can be found in the next section where discussion of adding applications to the cluster can be found.

7.2.2 Heartbeat Considerations

Please note the following differences of Linuxha.net in handling "heartbeat" packets compared to many other clustering products:

- The definition of which networks can be used for heartbeats is not explicitly defined - the software will make use of those it deems are reachable from the primary to the secondary server.
- At any one time only a single heartbeat package is sent across a single defined network, rather than being sent across all networks.
- If a heartbeat packet does not arrive in the "warn" time period the cluster will re-scan all networks looking for a suitable alternative.
- All heartbeats are sent over standard IP interfaces using the standard encryption method used for the cluster - no dedicated serial links or network interfaces are required.

This approach has been taken since it provides a good level of protection with a low CPU usage, and yet gives comparable fail-over times in most circumstances.

Hence the suitability of a network if the existing one fails is determined by Echo requests - the cluster supports ICMP, UDP and TCP echo types, though in most cases ICMP works well.

To guard against network partitioning more than one logical network should be defined in the cluster topology. Even if no applications are ever to use a particular interface and this interface is not also used for DRBD traffic, then still considered adding a network definition for it.

One novel idea that has been successfully used is to define a network consisting of a wireless card on each server. Although the cluster network daemon is not able to monitor this network it does provide a useful fail-safe in case the cluster network infrastructure dies.

The most secure way of assuring that a network is suitable for hosting the heartbeat requests is to make use of "UDP" or "TCP" packets. Neither of these are used by default since typically they require some configuration in "Inetd" or "Xinetd" on each host will need to be updated to ensure the relevant service is available.

For example in the "Inetd" configuration file, "/etc/inetd.conf" the following line was edited back into the configuration to support "TCP" based pings:

```
echo stream tcp nowait root internal
```


After making this change on both hosts refresh the Inetd daemon as well:

```
# killall -HUP inetd
```

Failure to ensure that the relevant Echo request referenced in the cluster configuration file will result in unpredictable cluster behaviour.

If the machines in question are linked to public networks then it is recommended that either a firewall is put between the servers and this network, or IP tables configured to each of the servers to ensure that the Echo requests are dropped unless they come from either cluster node.

7.2.3 Cluster Network and Locking Support

Although the setting defining the network ports for the network and locking daemons are actually optionally, there is no reason for **not** adding them, and lots of reasons for adding them.

When running a cluster with only a single application there is no benefit to running the locking service since it is designed to present two applications from interfering with resources that might be common. However since the locking daemon actually sleeps when not in use, it consumes virtually no CPU and thus letting it run in all scenarios is recommended.

Each node runs a locking daemon since the resources that are being guarded are local to each node. The locking daemon is very flexible - it has a built in list of locks that can be managed, though it is possible to define an alternative set, or override the existing lock time-outs if desired.

By default a sample locking configuration is included in the Linuxha.net distribution with the following path/name:

```
/etc/cluster/cllocks.xml
```

The file is straightforward XML:

```
<?xml version="1.0" standalone="yes"?>
<cllocks>
  <lock name="NET"
        desc="Network Config"
        maxt="40"
  />
  <lock name="NBD_CLIENT"
        desc="DRBD Primary Access"
        maxt="45"
  />
  <lock name="NBD_SERVER"
        desc="DRBD Secondary Access"
        maxt="45"
  />
</cllocks>
```

7.2.4 Building the Cluster

Once the "clconf.xml" file has been created on the primary node - "ServerA" in this example – the cluster can actually be "built", using the "clbuild" command, which has the following syntax:

```
# clbuild [-V|-verbose] [-F|--force]
```

The "clbuild" utility performs a large range of environment checks, (making use of "ssh" as appropriate to communicate with the other node in the cluster). It is recommended that the administrator run it with the "verbose" option to gain an understanding of the work being performed. The other point to consider is that this can be some time (up to a minute or more on large configurations) so the "verbose" option provides useful as a progress indicator as well.

Consider the following list which defines just some of the checks that get performed:

- Check that the /etc/cluster/clconf.xml file exists.
- Check that the cluster is not already running (unless the "--force" option is in use).
- Check that a build checksum file does not exist, (unless the "--force" option has been selected).
- Check that it is owned by root (0), and has no read access to anyone else.
- Check that the configuration file contained minimum number of values.
- All mandatory options which can not be defaulted are present in the configuration file.
- Check to see if the port pool contains at least 10 ports.
- Issues a warning if the "maxblockdevs" setting is too small (less than 10 entries).
- Checks to see if the DRBD kernel module exists and is loaded - attempts to load it if not.
- Ensure that LVM v1 or LVM v2 exist on both servers – and then if the necessary administration commands exist.
- Check to see if command line tools for network monitoring exist (which are used as a fall-back), and issue warning if not.
- Check that we are able to resolve the names of both nodes.
- Check that we are able to "ssh" to specified node via the node name.
- Check that the remote node can "ssh" back from the other node to the local node.
- Check that the nodes are able to ping the IP addresses defined for the cluster topology, and issue warnings if not.
- That the specified network interface cards defined in the topology currently exist.
- Ensure no card is referenced more than one in the topology.
- Ensure the network name defined for the DRBD replication data is defined for both nodes.
- Check that the current node is not already running the "cldaemon" daemon (unless the "--force" option has been used).
- Check that the the DRBD command line tools necessary are installed in the expected locations..
- Check that both nodes have user-space tools for the Logical Volume Manager.
- Check that the name of local machine is one of the node names in the configuration file.
- Check the interfaces defined for the local machine actually exist, (though some might not currently have IP addresses - so called "standby" cards)

Once SSH communication has been verified it will also perform further checks on the remote node, including:

- Check that the node is not already running the "cldaemon" daemon (unless the "--force" option was used).
- Check that the node does not already have a clconf.xml file (again unless "--force" was used).
- Check for the relevant DRBD and LVM user-space tools are present and available in known PATH locations.

Once these verification steps have been completed successfully it will:

- Create the /etc/cluster directory remotely if it does not exist
- Use the MAKEDEV.pl utility to create any DRBD device files if necessary on both nodes if necessary.
- Copy the configuration file to /etc/cluster/clconf.xml remotely
- Create suitable log directories on both servers if required
- Create resource files on both servers (under /etc/cluster/.resources) if necessary
- Create a checksum file for the cluster build.

Running the “clbuild” utility whilst an existing cluster is running **is supported** – When such an action takes place resources can be increase, and certain parameters will be dynamically changed for the running cluster. See the section covering Cluster Administration later in the document for further details.

“Udev” /dev configurations are also supported for Linux 2.6 based distributions. In such cases it will also create suitable entries in the “devices” directory to ensure they copied into “/dev” when the machine boots.

7.2.5 Creating “resource” flags

The program will create the following file for each currently available port defined in the range:

```
NNNN.free
```

This file will be created in the following directory:

```
/etc/cluster/.resources/ports
```

The above name indicates that the specified port is not currently allocated for any purpose in the cluster. If a port is assigned to serve as a part of a DRBD device (one for each file system is required), then the file would be renamed as follows:

```
NNNN.application.volumegroup.logicalvolume
```

For example:

```
9901.apache.datavg.data1
```

Notice that the port allocations for the same file system (hence logical volume / volume group) must make use of the same values - the cluster allocation of resources ensures this is the case, otherwise the cluster will not function.

Similar to the network ports, the actual block devices minor numbers are stored in the following directory on each host:

```
/etc/cluster/.resources/drbd
```

In a specified minor number (always starting at 0) is not in use it will have an entry in this directory of the following format:

```
NN.free
```

If this is allocated to a logical volume, in a volume group, for an application the form of the name will be as follows:

```
NN.application.volume.logical
```

By running “clbuild” the “free” resource files will be created – though if a resource is already in use, it is left unchanged – ensuring that if applications have been built the information regarding their resources is not lost. This is particular important since the cluster now supports re-running the “clbuild” command at any time.

7.2.6 Running the "clbuild" Utility

At that point the basic details of the cluster have been created - it will not run anything and no applications have been added / installed. However we have validated that both nodes are capable of running cluster software based on what hardware they have and the software installed.

When run in verbose mode the output of "clbuild" will look similar to the following - making it very easy to detect problems:

```
# clbuild --verbose
WARN 08/07/2005 05:04:19 This linuxha software has only be tested using the following
WARN 08/07/2005 05:04:19 range of Perl versions : 5.6.1 , 5.8.2 , 5.8.3 , 5.8.4 ,
5.8.5
WARN 08/07/2005 05:04:19 Running it with different versions of perl is not
recommended.
INFO 08/07/2005 05:04:19 Checking for required global entries
INFO 08/07/2005 05:04:19 Optional dataprotocol setting validated.
INFO 08/07/2005 05:04:19 Checking global Port Pool details
INFO 08/07/2005 05:04:19 Checking global Maximum block devices
INFO 08/07/2005 05:04:19 Checking configuration file version information
INFO 08/07/2005 05:04:19 Checking data detail value (and dependent required values)
INFO 08/07/2005 05:04:19 Checking type of data required
INFO 08/07/2005 05:04:19 Global section configuration validation complete
INFO 08/07/2005 05:04:19 Checking cluster timing details
INFO 08/07/2005 05:04:19 Checking node section details
INFO 08/07/2005 05:04:19 Checking IP address resolution
INFO 08/07/2005 05:04:19 Checking ssh cabability between Primary IP addresses (node
names).
INFO 08/07/2005 05:04:20 Networks defined for servera: public
INFO 08/07/2005 05:04:20 Networks defined for servera: public
INFO 08/07/2005 05:04:20 Validated network IP for public on servera: eth0 -
192.168.0.130
INFO 08/07/2005 05:04:21 Validated network IP for public on serverb: eth0 -
192.168.0.134
INFO 08/07/2005 05:04:21 Validated unique network/interfaces for servera:
INFO 08/07/2005 05:04:21 Interface eth0 is on network 192.168.0.0
INFO 08/07/2005 05:04:21 Validated unique network/interfaces for serverb:
INFO 08/07/2005 05:04:21 Interface eth0 is on network 192.168.0.0
INFO 08/07/2005 05:04:22 Successfully copied network configuration to serverb.
INFO 08/07/2005 05:04:22 Validated data replication network is ok (public).
INFO 08/07/2005 05:04:22 Able to send ping to DRBD Ip address 192.168.0.130 for
servera
INFO 08/07/2005 05:04:22 Able to send ping to DRBD Ip address 192.168.0.134 for
serverb
INFO 08/07/2005 05:04:22 Node servera is not running a cldaemon process (good)
INFO 08/07/2005 05:04:22 node serverb is not running a cldaemon process (good)
INFO 08/07/2005 05:04:22 DRBD administration tools found on servera.
INFO 08/07/2005 05:04:23 DRBD administration tools found on servera.
INFO 08/07/2005 05:04:23 Found LVM v2 on servera
INFO 08/07/2005 05:04:23 Found LVM v2 on serverb
INFO 08/07/2005 05:04:23 LVM v2 command set appears to be installed on servera
INFO 08/07/2005 05:04:23 LVM v2 command set appears to be installed on serverb
INFO 08/07/2005 05:04:23 Physical network check library miitoollib found on servera.
INFO 08/07/2005 05:04:23 Physical network check library miitoollib found on serverb.
INFO 08/07/2005 05:04:24 Creating DRBD devices on servera...
INFO 08/07/2005 05:04:24 Creating DRBD devices on serverb...
INFO 08/07/2005 05:04:25 Created Port Resources directory on servera
INFO 08/07/2005 05:04:25 Created 99 port allocation files on servera
INFO 08/07/2005 05:04:25 Created Port Resources directory on serverb
INFO 08/07/2005 05:04:46 Created 99 port allocation files on serverb
INFO 08/07/2005 05:04:46 Created DRBD Resources directory on servera
INFO 08/07/2005 05:04:46 Created 50 DRBD allocation files on servera
INFO 08/07/2005 05:04:46 Created DRBD Resources directory on serverb
INFO 08/07/2005 05:04:58 Created 50 DRBD allocation files on serverb
INFO 08/07/2005 05:04:58 Transferring cluster build checksum to serverb
INFO 08/07/2005 05:04:58 Successfully copied clconf.xml to serverb
INFO 08/07/2005 05:04:58
INFO 08/07/2005 05:04:58 Clbuild has completed without errors or warnings
INFO 08/07/2005 05:04:58
```

At this point “Serverb” will also have a copy of the “/etc/cluster/clconf.xml” file, and the log directories will have been created on each server, if necessary.

It should be remembered that “building” the cluster using the above routine does not actually start anything on the servers – this can be checked by running the following command:

```
# /sbin/cluster/clstat
```

The “clstat” is the cluster status reporting tool (and is explained in detail later). However in this instance all you will see is:

```
ERROR 08/07/2005 05:08:08 Cluster md10cluster is not running.
```

If there are problems with the cluster configuration and the build is not completed then the following error will be shown by clstat:

```
ERROR 08/07/2005 05:08:25 The cluster configuration file
ERROR 08/07/2005 05:08:25 /etc/cluster/clconf.xml appears to have been changed
ERROR 08/07/2005 05:08:25 but the changes have not yet been validated.
ERROR 08/07/2005 05:08:25 Please run the clbuild(1M) command first before
ERROR 08/07/2005 05:08:25 running this command again.
ERROR 08/07/2005 05:08:25 Please note that if the cluster is already running
ERROR 08/07/2005 05:08:25 you will need to use the --force argument. This will
ERROR 08/07/2005 05:08:25 not affect running applications.
```

This occurs because the checksum of the last validated file is different from the contents of the existing configuration file. This mechanism is used by Linuxha.net to ensure that when changes are made to the configuration files the administrator must ensure the appropriate “clbuild” or “clbuildapp” commands are completed to validate the update configurations.

Once the configuration has been built successfully the cluster should be started to indeed validate everything is in place:

```
# clform
```

The above command should generate output similar to the following:

```
INFO 09/07/2005 23:50:04 Validated checksum for cluster configuration
INFO 09/07/2005 23:50:04 SSH communication to serverb will be:
INFO 09/07/2005 23:50:04 192.168.0.134 ("public" network)
INFO 09/07/2005 23:50:04 Checking that the cluster is not already running...
INFO 09/07/2005 23:50:04 *** ATTEMPTING TO FORM CLUSTER md10cluster ***
INFO 09/07/2005 23:50:04 Starting cldaemon on servera...
INFO 09/07/2005 23:50:05 Starting cldaemon on serverb...
INFO 09/07/2005 23:50:05 Waiting for cluster to form...
INFO 09/07/2005 23:50:11 Cluster md10cluster started successfully.
```

Notice that the utility attempts to work out an IP address to communicate with the remote node? This functionality is used throughout Linuxha.net - it ensures that even if the network path to the IP address defined for the node name is not available then the IP address associated with another defined network in the topology can be used for communication instead (either SSH or Linuxha.net client/server protocol).

Once the cluster has been started the administrator can confirm this by running “clstat” again, this time it should show instead:

```
Cluster: md10cluster - UP
```

Node	Status
servera	UP
serverb	UP

If the cluster does not the most likely reason is that the two nodes have a difference of more than 10 minutes. In this case the following will have been shown:

```
ERROR 08/07/2005 05:17:26 Cluster has failed to start. Log entries given below:
ERROR 08/07/2005 05:17:26
ERROR 08/07/2005 05:17:26 INFO 08/07/2005 05:17:23 Response to ECHO was FORMING (our
state=FORMING)
ERROR 08/07/2005 05:17:26 INFO 08/07/2005 05:17:24 Response to ECHO was STARTING (our
state=FORMING)
ERROR 08/07/2005 05:17:26 INFO 08/07/2005 05:17:24 Response to ECHO was UP (our
state=FORMING)
ERROR 08/07/2005 05:17:26 INFO 08/07/2005 05:17:24 Both nodes agree UP
ERROR 08/07/2005 05:17:26 ERROR 08/07/2005 05:17:25 Time difference between nodes is
>10 minutes - require --force to start!
```

As the error indicates if the administrators wishes to form the cluster without correcting the time difference, then run the "clform" again with the force option:

```
# clform --force
```

At this point the "clstat" command should indicate the cluster has been formed, and so the administrator can move on with the next step - adding applications to the cluster. The next section covers installation of a sample "Apache" application - for more generic guidelines covering any application please view the "Linuxha.net Basic Application Configuration Guide".

8 Building the sample "Apache" package

8.1 Creating the Application Configuration File

Applications can be added to the cluster at any time - the cluster does not need to be running. However if the cluster is up and functioning the application, once built, will be available immediately for use - there is **no** requirement to restart any software components.

The only requirement is that both nodes are contactable - this is because the process of building a application definition must define and allocate resources on both nodes in the cluster.

Before an application is made available via the cluster the administrator must define a file which describes the application in detail – including networking, storage and application script requirements.

The example here is based on a sample configuration that you can download if you wish – it does include a build script to automate much of this process. See the link on page for further details.

The configuration file can be placed on either node in the cluster – though if the application is already clustered and is running, the steps must be performed on the node that currently runs that application. Of course for a new application any node will do - though that node must currently have all the file systems that will contain data to be mounted.

This means that the administrator may have to define a volume group on both nodes, create new logical volumes on a primary node, create file systems and copy in the data to replicate - and then mount it - more details follow the description of the configuration file to generate.

The configuration file you must create for an application is as follows:

```
/etc/cluster/application/appconf.xml
```

In this instance the name of the configuration file created is:

```
/etc/cluster/apache/appconf.xml
```

The contents of this example configuration file are as follows:

```
<?xml version="1.0"?>
<appconf>
  <global>
    <version>0.1</version>
    <name>apache</name>
    <takeover>normal</takeover>
    <syncrate>2000</syncrate>
  </global>

  <networks>
    <network net="main"
      ip="172.16.177.200" netmask="255.255.255.0"
      checklist="172.16.177.1" checkpercent="100"/>
  </networks>

  <vg>
    <name>app01vg</name>
    <type>filesystems</type>
  </vg>

  <application>
    <startscript>/apache/admin/scripts/startapp</startscript>
    <stopscript>/apache/admin/scripts/shutdown</stopscript>
    <maxstoptime>10</maxstoptime>
    <maxstarttime>20</maxstarttime>
  </application>
</appconf>
```

Each of the entries will now be explained, starting with the global options. Please note that some directives are described in more detail later in the document - see the section “Easier Application Management”, starting on page .

Element	Purpose
global.version	The version of the application configuration file format – this will be used to ensure that upgrades to software notice different options available.
global.name	The name of the package that is being described. This should be the name of the directory where this file lives, and should not contain white space.
global.takeover	Can be either “normal” or “force”. If normal then the cldaemon will only allow a fail-over to a node if that node has current data, “force” will allow fail-over even if the data is not current. Note that if this item has not been specified, or specified with in incorrect value, it will default to “normal” (which maximises data consistency at the expense of availability).
global.syncrate	(Optional). The rate (defined in Kb/second) that should be used when synchronizing data between the two nodes. This value defaults to 1000Kb/second. This setting is per-device .
global.autostart	(Optional). Set to “true” or “false” (Optional). This is used by “clform” and “clstart” to indicate whether to automatically start the application when the cluster is formed.
global.dependencies	(Optional). A comma-separated list of other applications in the cluster that should ideally be up and running prior to this application starting. It is used by “clform” and “clstart” to determine the order of starting applications.
global.preferred_node	(Optional). A directive which defines where the application should reside if the cluster is currently running optimally.

The networking section basically defines which networks and the IP addresses to use. The entire section is optional, and the “network” element can occur zero, one or more times for an application. The attributes that are defined in each “network” entry are mostly optional:

Element	Purpose
<code>network.net</code>	The name of the network the specified information should be applied to.
<code>network.ip</code>	One or more IP addresses to place on the specified network. The format of the IP address is: AA.BB.CC.DD[, EE ...] Hence support for multiple IP addresses is currently possible by changing the last byte of the IP address.
<code>network.netmask</code>	(Optional). If the default network mask for the specified network is not suitable when assigning the IP addresses than the netmask can be specified.
<code>network.broadcast</code>	(Optional). Only necessary if the default broadcast address is not valid for the IP addresses being assigned.
<code>network.checklist</code>	(Optional). A comma separated list of host names or IP addresses that should be pinged to check IP-level connectivity. If not set no IP level connectivity checking will be available - even if configured for this application in Lems.
<code>network.checkpercent</code>	(Optional). The percentage of hosts that must be ping-able on a given interface to consider it working. If this is not specified it defaults to 100. Obviously the value should be between 0 and 100, where 0 effectively turns off IP monitoring.

Administrators of earlier versions of Linuxha.net clusters will notice that the network configuration is for application avoids any information on the overall cluster network topology. This change makes the cluster easier to manage, but was also necessary to introduce the dynamic changing of the clustered applications or the cluster itself.

The “checklist” and “checkpercent” are used by the cluster (if available from a currently running application), to check whether a new interface to use is actually working as expected. Mostly such information is not required in most clusters since the inbuilt physical network connection monitoring offered by the cluster network daemon is adequate.

The type of ping and the time-out to use when using “checklist” and “checkpercent” are defined in the cluster topology file.

All ping checks are carried out in sequence at present. Support for “parallel” pings (which can be much faster when using large timeouts) will be supported in later versions.

The remaining two sections, “vg” and “application” are also optional. The entries in the “vg” section can be repeated zero or more times, once for each volume group.

Element	Purpose
<code>vg.name</code>	The name of the volume group that is associated with this application. This does not need to include the <code>/dev</code> prefix.
<code>vg.type</code>	How to treat the volume group – currently the only accepted entry is “filesystems” - which means that all mounted file systems will be recorded and then mounted or unmounted as required when the application starts or stops.
<code>application.startscript</code>	The name of the program to run when the application needs to be started. This script MUST EXIT once the application has been started - thus it must “nohup” and “background” any non-daemon type applications as necessary.
<code>application.stopscript</code>	The name of the program to run when the application needs to be stopped. It should exit once the application has stopped. Please note that if this takes too long the package may decide to continue the package stop without waiting for this to complete.
<code>application.maxstoptime</code>	The maximum amount of time (in seconds) that the cluster will attempt to stop the application for before aborting the process and continuing with the steps necessary to stop the application.

Current versions of Linuxha.net support the “startscript” and “stopscript” have parameters, which must be white space separated from the command path.

8.2 Network Availability Considerations

Although the current versions of Linuxha.net no longer support “bonding” to combine multiple interfaces, such functionality is likely to return in some form in later releases. At the moment the administrator can skip this section, since it now only contains useful information, it does not affect how the cluster can be configured presently -hence skipping to page 59 is recommended.

8.2.1 Bonding verses Fail-over network Types

These two different types of network functionality for provision of the application IP address are available since they meet different requirements for high availability clusters. The differences between each type will can be summarised as follows:

- IP availability on NIC/Cable failure
Most applications and clusters will make use of the “fail-over” network type. In this scenario hardware level checking, (if supported by the network cards), will typically spot a problem within two seconds and migrate the affected IP address to another candidate card (or switch node if deemed more appropriate).

Hence there will be a small outage (several seconds) whilst the IP address is migrated to another logical interface. If multiple applications are running this outage might last longer for some applications compared to others since each application will be done in turn.

When using the “bonding” type of network the driver itself is responsible for checking the health of the network connections and on a hardware failure will deal with the issue transparently to the application or cluster software. The result is that no network outage will occur - obviously a significant advantage compared to “fail-over” network types.

- Performance
When using “fail-over” networking the bandwidth available to an application is limited to that of the current card in use. The “bonding” driver, depending on the environment and chosen configuration, allows bandwidth aggregation of all interfaces for the application, thus offers a considerable advantage for applications expecting a high volume of client/server traffic.

However in most circumstances a single card, (particularly if Gigabit), is adequate is everything apart from the most unusual of circumstances.

- **Complexity**
The "fail-over" network type is very straightforward - simply provide it with a list of cards that are connected to the correct network for the application and it will make use of them, performing hardware and/or IP level checks as defined by the application configuration.

Unfortunately channel bonding as used by the "bonding" type is much more complex - it offers many modes of functionality, including active/passive or active/active configurations. Further the functionality available is determined by the cards in use and the network equipment they are attached to.

The default configuration used by Linuxha.net should suit most environments, but is built only for availability rather than aggregation of bandwidth.

If channel bonding is chosen it is recommended that the administrator read carefully the nodes in the following files, for whichever version of Linux is in use:

Documentation/networking/bonding.txt

Please endeavour to manually try a particular configuration before attempting to configure an application to use it!

- **Resource Requirements**
With network types of "fail-over" the IP address is assigned as a virtual IP address on the specified interface. This means that the same interface can be used for multiple applications simultaneously - very useful if more than a single application is clustered or a limited number of Ethernet cards are available.
Due to the close interaction with the underlying hardware such an approach is not possible when using the "bonding" network type. In this cases the cards defined for the application must not be in use at all. If they are then the cluster software is not able to define a bonding device making use of them and the application will fail to start.

Hence "bonding" is only really useful if you allocate particular cards to each application - only possible if the administrator has one or possibly two applications to cluster, and a significant number of network cards!

8.2.2 Supported Bonding Modes

The default mode should work under all circumstances, however the "mode" value can be changed to other values also, such as:

Mode	Characteristics
0	Basic round robin policy - provides fault tolerance and performance by using different cards to transmit each frame. Requires no specific protocol support from the switch, and indeed the interfaces can be connected to different switches, hubs or directly to the other node.
5	Adaptive transmit load balancing - again this does not require any particular protocol support from the switch. One slave interface is responsible for retrieving all traffic, (obviously changing if it fails), whilst all transmissions make use of the least heavily used interface, taking into account the performance available for each card. This policy can be useful if the amount of traffic sent from the application is much larger than the amount the clients send, (i.e. client-server rather than peer-to-peer).
6	Adaptive load balancing - like mode 5 but also actively balances incoming traffic across the available interfaces, and again requires no special switch support. The Ethernet cards when used in this mode must support setting the hardware address, since load balancing is achieved using ARP negotiation.

When “bonding” is used at least two interfaces must be specified for each node. When the application starts on a server a check is made against the list of interfaces to find two that are not currently being used - these are then bonded for the duration of the running of the application. Of course when the application is restarted on the same server the same two cards might not be used.

The software determines that a card is a candidate for bonding if the IP address is currently 0.0.0.0 - hence the interfaces specified in the configuration should be set to this IP address as part of the boot sequence on the host.

8.2.3 Link-level checking verses IP level checking

Although IP level checking is supported the recommended approach is to ensure that the network cards support link level checks - most do these days. In this case the software will make use of the facility to ensure that the network connection for a card is working at the link-level, which is much more reliable than simply pinging lots of IP addresses.

However IP address level checking is supported since some cards do not support such checks. It is also possible, (and recommended in some circumstances), to use both at the same time.

With IP level checking the range of IP addresses specified is pinged on a regular basis, and if the percentage of successes drops below the value specified in the application configuration file, then the cluster will attempt an IP fail-over.

8.2.4 Sharing Multiple IP addresses

You may wonder whether you need to have as many Ethernet cards as applications you intend to run ... luckily this is not the case when using the “failover” IP configuration type! Each application that is run, after choosing a particular Ethernet card will create an “IP alias” on that card. This means that several applications can be using the same Ethernet card to advertise their individual application IP addresses without problems.

In most circumstances the recommendation is to use at least two network interfaces with connections to the client networks.

8.3 Checking the Application Configuration

The first step in using the application in the cluster is to check the configuration specified for the application is correct – for this we use the “check” mode of the “clbuildapp” program.

For applications that need to be defined in the cluster the commands should be run on the node where the application file systems, in the volume groups specified (if any), are currently mounted.

A typical invocation of “clbuildapp” might be:

```
# clbuildapp --application apache --verbose --check
```

The flags used in the above command should now be obvious! Please ensure the “--check” option is included to indicate that no actual steps to build the application are to be performed - just a validity check.

The “clbuildapp” utility supports the following command line arguments:

Flag	Purpose
--application	The name of the application to check, build or synchronise.
--verbose	The utility will report informational messages as it goes about whatever business it needs to. All lines will start with “INFO” and will appear on the standard output device.
--check	Only perform any validation checks that the utility would normally do - do not take any actions to change either node.
--vgbuild	Check the volume group configuration is the same on both nodes, which may lead it to automatically build the required logical volumes on the remote node in some circumstances ³ .
--build	Allocate all resources required to allow the application to function – this includes network ports, RAID devices and NBD devices, as well as file system mapping information.
--force	Force the step – necessary if it appears that the package has previously been successfully built.

Before running the “clbuildapp” routine the following list recaps the necessary steps that are expected to have been taken:

- The required volume group has been created on the remote machine.
- All necessary logical volumes on the local machine have been created with the correct sizes, and the file systems on these logical volume mounted (including use of the correct mount options).
- The mount file systems are populated with the data that is to be replicated.
- The application has been tested, but is **not** currently running (i.e. the file systems are mounted, but no processes are using them for open files, or as a current working directory).
- The cluster has been built as described previously with the “clbuild” utility – see page for details.
- Although the logical volumes do not need to exist on the remote system, the volume group must contain enough free space to create them if they do not. In all cases the file systems on the remote node **should not** be mounted.
- The IP addresses associated with the application should not be available, either defined on the local machine, or exist at all on the network.

The “check” stage is just the first of four stages that must be performed to move from an application defined on local storage on the current node, to one that exists in the cluster with a true replicated data copy. In summary the steps are;

- *Check* – This checks the configuration for any errors, omissions or potential problems that might occur.
- *Volume Group Build* – checks to see if the volume groups in question on the local server are mirrored on the remote server, creating logical volumes if necessary under certain conditions. It will also ensure all meta-data volumes for DRBD are created as necessary on both cluster nodes.
- *Build* – the necessary resources to run the application in the cluster are allocated, and if the cluster daemons are running the package will be registered with the running cluster.
- *Synchronise* - The local file systems are unmounted from the native mount points and then the data contained within them is replicated to the remote node.

³In this case the volume group must already exist and contain enough free space for the logical volumes to create.

Each of the above stages will be shown and described in turn. Firstly consider the "--check" option. Again a large number of checks to validate the environment are performed, including:

- Check that the specified application is not already running in the cluster (unless used with the "--force" option).
- Check that the "/etc/cluster/clconf.xml" file exists and has the correct permissions (to ensure it is only readable by "root").
- Check the node details to ensure one node list is from the /etc/cluster/clconf.xml file
- Check the remote node is available via SSH via any defined topology IP address.
- Check to ensure that a directory "/etc/cluster/.resources/fsmap/application" **does not** exist locally - and if it does only continue if "--force" has been specified.
- Check that the local and remote nodes can the md5sum command
- Check that the /etc/cluster/clconf.xml matches on the remote node, (via the use of md5 checksums) - this ensures clconf.xml correctly defines a cluster.
- That the DRBD kernel module is present on both hosts.
- Check that the "/sbin/cluster/utlils/mkdir" utility exists on the remote node, (used to create mount points for file systems - more later).
- The "appconf.xml" for this application can be read in and parsed as valid XML.
- Check that the version of the configuration file matches the version of the "clbuildapp" program.
- If network details are given check the networks exist in the cluster topology.
- If volume group details exist check that they exist on the local and remote machines.

Please note that when the "--force" option is used the "clbuildapp" can be used in "rebuild" mode - that it is can refine an existing application that is already clustered. This can even be done whilst the application is running. Later sections of this document described how such features can and should be used.

To check the application configuration, simply run the following command:

```
# clbuildapp --application apache --verbose --check
```

When running in verbose mode the output generated will appear similar to the following:

```
sample clbuildoutput here
```

Before continuing any problems indicated must be fixed - if not the other stages will abort with an error since the required checksum to indicate a checked configuration will not be present, or will be incorrect.

8.4 Validation / Build of Volume Groups

The next step of the build process is to ensure that the volume groups (if any) for the application are the same on both sides of the cluster, (they need to have the same named volumes, of the same size, in the same volume group).

This step will also ensure that the meta areas for DRBD data replication are created. For each file system to replicate a separate 128Mb logical volume, on both nodes, is necessary. These are created automatically during this step if they do not yet exist. If space is not available to allocate these volumes, (which are named "lv_meta" - where "lv" is the name of a particular logical volume to replicate), this this phase of the application build will fail.

To use the "clbuildapp" tool to validate / replicate the volume group configuration on the remote node, the "--vgbuild" argument is required. Before the administrator runs it please remember the following point:

Please ensure that the volume group, at least, has been created on the other node – and that it is large enough to contain the logical volumes in the cluster, including the logical volumes used for DRBD data replication.

Again use of the “--verbose” option is recommended. In this instance the command and sample output would be (omitting duplicated lines from the “--check” run):

```
# clbuildapp --application apache --verbose --vgbuild  
. . .  
--vgbuild output
```

Following the successful completion of the command the volume groups on both nodes may have been changed. Use the following command to view the output on each to view the changes.

```
# vgdisplay -v app01vg
```

Since the “--vgbuild” option used the list of logical volumes defined on the current host and replicates them remotely. If the application already exists and is running the semantics are somewhat different. See sections on cluster administration later in the document for details.

8.5 Allocating Application Resources

At this point the logical volumes will have been validated across the two nodes – the next step is to actually allocate the remaining resources that are needed to allow the application to run in the cluster. This is done again using the “clbuildapp” command – this time using the “--build” flag. When this argument is used the following steps are carried out:

- Check to see if the “/etc/cluster/.resources/fsmap/*application*” exists and unless “--force” is specified then the program should abort (since it appears that the application is already defined in the cluster).
- Check to see if the “--build” checksum file exists, and again require the “--force” option if it does to ensure that application does not overwrite the existing configuration.
- Check to see if an application of this name is already running in the cluster, (if the cluster is running). If it is abort unless “--force” has been specified.
- Check that all volume groups mentioned in the configuration file exist and are present locally.
- Check that all the volume groups mentioned are present remotely.
- Check that all logical volumes mentioned are active, (since we're getting the information from /proc they will only show up if active!)
- Check that all logical volumes are open on the local machine.
- Check that all open logical volumes are mounted locally - (necessary to understand the mount point information for the cluster, and the type of volumes).
- Ascertain the number of ports and DRBD devices needed for this application and ensure enough resources for each exist on both nodes.
- Remove any existing “TIME” file for this application if they exist, (these are created when an application starts on a node).
- Check to ensure that all logical volumes on the remote node are closed.
- Ensure that the mount points specified locally can be made on the remote machine if they do not exist.
- Allocate the necessary port and DRBD devices for this application, (both locally and remotely). If the devices for already exist for these resources then simply use those.
- Copy the application configuration files across to the “/etc/cluster/*application*/appconf.xml” on the remote node.
- Generate the fsmap directory entry for this application locally, (including mount options).
- Copy the fsmap details to the remote node, (see the “resources” section next for details of this file).
- Calculate and write a copy of the build checksum to both servers – indicating the current application configuration is valid.

The command, as well as some typical output from running this command are shown below, again use the “--verbose” option.

```
clbuildapp --application apache --verbose --build
. . .
output of --build stuff here
```

8.6 Synchronising the Cluster File systems

At this point all resources required for the application as part of a Linuxha.net cluster are in place. However at this point the logical volumes on the remote node will not have the same contents as the local machine. Although the application could actually be used, the recommended approach is to explicitly synchronise the data first, and once complete consider starting the application in question.

This synchronisation effort can take place whether the cluster software is running or not, and if the cluster is running, whether other applications managed by the cluster are running or not.

The synchronisation can be performed using yet another flag available from the “clbuildapp” command; “--sync”. This option essentially performs the following actions:

- Get a list of volume groups and logical volumes using the “fsmap” entry details for the application.
- Check to ensure that the application is not already running (if the cluster appears to be running).
- Validate that the build checksum matches the contents of the application configuratoin file, aborting if not.
- Ensure that all file systems on the volumes contained in the “fsmap” are currently unmounted - and if still mounted, attempt to unmount them – and aborting if this is not possible.
- Call the “/sbin/cluster/utlis/drbd_tool” remotely to start the DRBD device server processes.
- Call the “sbin/cluster/utlis/drbd_tool” locally to start the DRBD device clients.
- Force the local copy “primary”.
- Force a full synchronisation if “--forcesync” option has been included (see below).
- Monitor the status of the DRBD devices and wait until all devices for this application are synchronised.
- Stop all DRBD devices on local host (for this application)
- Stop all DRBD devices on the remote host (for this application)

The command also supports an option “--forcesync”. This can be used to force data synchronisation to take place, even if the DRBD device meta data indicates this is not necessary. See “Managing Application File systems” section below for typical instances when this might be necessary.

An example output of a build session is given below, (with some output removed to save space/time/boredom):

```
clbuildapp --application apache --verbose --sync
. . .
sample --sync output here
```

The time taken for this to run really depends on the volume of the data to synchronise, the performance of the machines, and the performance of the network connection between them. Typical commodity PCs with a low cost 1Gb/sec network cards should be able to achieve 20MB/sec or more to standard PATA or SATA disks.

A common problem that occurs is that the rate of synchronisation appears to be too slow. The administrator should recall that the “syncrate” application setting only defaults to 1MB/sec per file system - this value can be increased and the application “rebuilt” if necessary.

8.6.1 Managing Application File systems “outside” the cluster

Once the data synchronisation is complete, all file systems should only be mounted via the DRBD devices, rather than directly using the volume group and logical volume. If the cluster is not running then the application can be still be started in most instances, but even if this is not the case either of the following approaches is possible:

- Manually start the DRBD devices (even if just locally)
The “drbd_tool” (manual page available) allows the DRBD resources for an application to be made available on the local machine. For example to start the DRBD devices for the “apache” application the following commands could be used:

```
# /sbin/cluster/utlis/drbd_tool --application apache --action=start
# /sbin/cluster/utlis/drbd_tool --application apache --action=mount
```

At this point any changes could be made and the administrator should then run the following once complete:

```
# /sbin/cluster/utlis/drbd_tool --application apache --action=unmount
# /sbin/cluster/utlis/drbd_tool --application apache --action=down
```

This ensures the meta-data locally is updated. When the application is started in a normal manner as part of the cluster it will resynchronise the changes partitions of disk automatically.

If the remote host is still available then the remote DRBD could be started as well, thus ensuring all data remains synchronised prior to the application starting in the cluster. To do this an additional command is necessary before any after making the changes:

```
# ssh remote /sbin/cluster/Utils/drbd_tool --application apache \
  --action=start --nopriary
# /sbin/cluster/Utils/drbd_tool --application apache --action=start
# /sbin/cluster/Utils/drbd_tool --application apache --action=mount
```

Following any changes run:

```
# /sbin/cluster/Utils/drbd_tool --application apache --action=unmount
# /sbin/cluster/Utils/drbd_tool --application apache --action=down
# ssh remote /sbin/cluster/Utils/drbd_tool --application apache \
  --action=down
```

➤ **Change local file systems and force synchronisation**

The alternative approach, which is strongly discouraged, is to mount the file system directly. For example:

```
# mount -t jfs /dev/app01vg/lv01 /apache
```

In such cases following the change the file system should be unmounted and a complete synchronisation forced:

```
# umount /apache
# clbuildapp --application apache --verbose --sync --force --forcesync
```

The “--forcesync” is absolutely necessary to ensure the local file system contents overwrite the remote one - since DRBD is unaware of the changes the administrator will have made to the local copy.

However before attempting to use either of the above options the administrator is encourage to use the preferred approach - via the “--force” option to the “clstartapp” command to force the application to start if possible, make the alterations, and then force it to stop. These commands work in many cases, even if only one node is available, or no cluster daemons are currently running.

8.7 Understanding Application Resources

When a package is built or checked the resource files are called into play. It should be noted that if the package has any volume groups, (these actually optional), then network ports, and DRBD devices will be allocated during the creation of the package, (more accurately during the “build” stage).

If all resources have been allocated correctly the following file will also have been created on both nodes:

```
/etc/cluster/.resources/fsmap/application
```

This file is used to provide information on file system types, mount points, mount options and logical volumes for all files in the package. It also contains the current size of the logical volume (to track changes to volumes, see later in the document for details). It is created first on the local node and then copied to the remote node during the application build process described previously.

The format of the entry on the “fsmap” directory is as follows:

```
<Volume group>:<Logical Volume>:<Mount Point>:<fs type>:<mount options>:Size
```

For example you might get the following typically:

```
app01vg:clcst:/cluster/control/myapp:reiserfs:ro:10240  
app01vg:home:/myapp/home:ext3:data=ordered:20480  
app01vg:data:/myapp/data:reiserfs:rw,notail:102400
```

Notice that this information is necessary to both “fsck” and “mount” the file systems when they are used by the cluster – though this information is generated during the build and can not be assumed. This means that changing the file systems in the cluster does require some reconfiguration - though the standard “clbuildapp” utility typically caters automatically with situations when this file is required to change.

If the “fsmap” file for an application does not exist, then the “application” has failed to build successfully. The reason for this will have been shown during the build phase previously described. Without this file, the application can not be started under Linuxha.net.

9 Application Configuration and Monitoring

As stated in the introduction it is important that changes to the state of the cluster and application and monitored and reacted to - otherwise the purpose of the clustered software is called into question! Monitoring takes place via the following mechanisms:

- Cluster Daemons
- Cluster Network Daemon
- Application "Lems" Daemon

From the administrator's viewpoint the key requirement for monitoring the application is ensuring that a configuration for "Lems" is suitable for that environment. This section covers a basic implementation of a sample "Lems" session for the sample "Apache" configuration, for more detailed technical implementation information, please see the section beginning on page .

However before discussing any monitoring the scripts necessary to start and stop the application must be covered in a little more detail. In the "appconf.xml" file shown earlier the "application" section describes the commands used:

```
<application>
  <startscript>/apache/admin/scripts/startapp</startscript>
  <stopscript>/apache/admin/scripts/shutdown</stopscript>
  <maxstoptime>10</maxstoptime>
  <maxstarttime>20</maxstarttime>
</application>
```

Notice that these scripts can exist outside of the replicated file systems, or actually as part of the file systems covered by the cluster. Where they are kept depends on the administrator and the application being clustered. Some applications expect files to exist in certain directories, such as "/etc". In such cases a replicated location is not feasible. However with a replicated solution only a single copy is kept – preventing problems with nodes having different scripts, which might result in subtle errors.

These scripts are usually simple shell scripts, though can be as complex as desired. The exit code of the "start" script is important – it must be 0 if the application has started successfully. If this is not the case then the cluster software takes the result as a failure and the start up of the application **will be aborted**.

The contents of the "start" script in this instance is very basic –just a call to the standard "apachectl" script - which is expected to be in the same directory, rather than a more usual location.

```
#!/bin/bash

/apache/admin/scripts/apachectl start
exit $?
```

Of course stopping is just as straightforward – the "shutdown" script is basically the following:

```
#!/bin/bash

/apache/admin/scripts/apachectl stop
exit $?
```

9.1 Start/Stop Script Interface Requirements

The start and stop script are **not** called with any arguments by default. If arguments are required then they should be added directly after the command in the XML configuration file for the application.

To expand on the information previously given the following table should be considered when setting the return code for either script.

Script	Return Code	Consequence
Start	0	The application is thought to have been started successfully, any remaining steps are performed and the application successfully starts under Linuxha.net.
	1	The application start-up has failed. If a forced application start-up is not in effect then the start-up of the application will fail. For a forced start-up a warning is issued, though the process of making the application start under Linuxha.net will continue.
Stop	0	The shut-down of the application silently continues.
	1	Am warning is given, but the application shut-down continues.

When stopping the application the return code is recorded if non-zero in output, but no other actions take place. This is because the shut-down will ensure any processes using the clustered file systems are killed anyway.

The other important aspect when writing or using scripts is to ensure the start and stop **times** are configured to allow the script to do what it needs to do under normal working conditions. Initially it is recommended that the start and stop times are set very high so they will never be crossed. The administrator should then refine them (reducing them in most cases), to be 50% - 100% longer than the application typically takes to start or stop.

This is because once this time is exceeded the application start-up and shut-down of the application perform the following actions:

- A warning is issued after 1/2 of the available time to start or stop the application has elapsed.
- A warning is issued again at the limit of the time configured to start/stop, at this point a HUP signal is sent to the process that was starting/stopping the application.
- If the application does not start/stop for 5 seconds after sending the HUP signal then a KILL signal is sent, effectively stopping the application start-up or shutdown.

Whether shutting the application down or starting it up at this point is irrelevant; Linuxha.net will unmount the replicated file systems for the application, killing off any processes using them if necessary.

9.2 Providing a "Lems" Monitor for the application

Now that the application environment is ready the administrator must create a "Lems" session which detects and responds to events that affect just this application, or are most relevant for this application. Detailed information on using Lems is found later in the document (see page 111). Also technical configuration information on Lems can be found starting on page 174.

For the sample "apache" application (also distributed as part of the "linuxha_apache" package), the administrator will find the following file:

```
/etc/cluster/apache/lems.local.xml
```

The contents of this file are mainly a series of "check" entries similar to the following:

```
<check>
  <name>ip</name>
  <type>internal</type>
  <module>ip_module test01</module>
  <interval>10</interval>
  <action_list>
    <action rc="0" action="NOP"/>
    <action rc="1" action="RUN move_ip"/>
    <action rc="2" action="STOP"/>
  </action_list>
</check>
```

For comprehension of the environment the the only entry that should currently concern the administrator is the one with a "name" element of "httpd", which will appear as follows:

```
<check>
  <name>httpd</name>
  <type>internal</type>
  <module>procmon /etc/cluster/apache/httpd.xml</module>
  <interval>10</interval>
  <action_list>
    <action rc="0" action="NOP"/>
    <action rc="1" action="STOP"/>
    <action rc="2" action="FAILOVER"/>
  </action_list>
</check>
```

The "module" setting for this entry indicates that a module called "procmon" will be called every 10 seconds. The "procmon" module is used for process monitoring and typically at least one entry similar to the above will be present in a Lems configuration file for a clustered application – assuming the administrator wishes to include application level checking / re-start / failover functions. This is not strictly necessary, but is sensible for most applications that are clustered.

9.3 Sample Process Monitor Implementation

Lems works on the principal of modules - each module performs a different type of check (for example the configuration shown in the previous section made use of "procmon" - the process monitor module).

Several "Lems" configuration modules are provided as standard to handle many of the "system" related activities that may impact the availability of the application or cluster. The ones that are "cluster-wide" such as monitoring swap space are only really suitable for environments that have a limited number of applications - and in such circumstances if such a monitor is required to be run, it should only be run for a minimum number of applications to reduce system overhead.

For the sample Apache application, the process monitor entry in Lems requires a simple configuration file to be built describing the processes to monitor and actions to take. This XML file is typically placed in the following directory on each node in the cluster:

```
/etc/cluster/application/name.xml
```

The name of the configuration file is left to the administrator, though in this example it is called "httpd.xml" and has the following contents:

```
<?xml version="1.0"?>
<procmon>
  <global>
    <logdir>/var/log/cluster</logdir>
    <restarts>3</restarts>
    <resetwindow>3600</resetwindow>
    <restartcmd>/apache/admin/scripts/restart</restartcmd>
  </global>
  <process>
    <label>Web Server</label>
    <user>nobody</user>
    <process_string>httpd</process_string>
    <min_count>1</min_count>
    <max_count>10</max_count>
  </process>
</procmon>
```

Full details on the meanings of these fields are shown later when the "Lems" environment is described in much greater detail (starting on page 111). However in essence the above will run the "restart" script for the application if less than one process or more than 10 processes for the user "nobody" contain the string "httpd" are not found or found respectively.

The other entries in the sample configuration require no customisation or further configuration files. The administrator needs take no further action – the Lems daemon for a particular application is started by the Cluster daemon when a package is started, and stopped when it is no longer required.

The cluster daemons also check to ensure that these monitors continue to run, and if the process dies it is re-started.

The administrator is responsible for checking the cluster configuration files for excessive restarts of the Lems daemon for an application. This would indicate an underlying issue with either Lems or the configuration for that application which should be investigated as soon as possible.

The Lems daemon for a particular application typically is configured to run in verbose mode, and will use the following log file typically⁴:

```
/var/log/cluster/lems/lems-application.log
```

This log file should be first as the first action if there are any problems running a particular application. Typically this log records significant information, though it does require a “verbosity” setting – though this is enabled by default.

Alternatively if is possible to run the command in “check” mode from the command line, (assuming the package is not running), with a command such as:

```
# lems.pl --application apache --config \  
/etc/cluster/apache/lems.local.xml --verbose --check
```

The output of this command would look something similar to the following:

```
INFO 12/06/2005 22:58:06 Using modules from : /sbin/cluster/lems/modules  
INFO 12/06/2005 22:58:06 Using programs from : /sbin/cluster/lems/programs  
INFO 12/06/2005 22:58:06 Writing logs to : /var/log/cluster/lems  
INFO 12/06/2005 22:58:06 Validating port not open locally...  
INFO 12/06/2005 22:58:06 Validating port not open on sl4s2...  
INFO 12/06/2005 22:58:06 No clash for application samba - uses port 8801 not 8800.  
INFO 12/06/2005 22:58:06 Listening on port : 8800  
INFO 12/06/2005 22:58:06 Global initialisation complete.  
INFO 12/06/2005 22:58:06 Started local server on port 8800  
INFO 12/06/2005 22:58:06 Validating monitor entry fsmonitor...  
INFO 12/06/2005 22:58:06 Validated monitor entry fsmonitor successfully.  
INFO 12/06/2005 22:58:06 Validating monitor entry httpd...  
INFO 12/06/2005 22:58:06 Validated monitor entry httpd successfully.  
INFO 12/06/2005 22:58:06 Validating monitor entry flag_check...  
INFO 12/06/2005 22:58:06 Validated monitor entry flag_check successfully.  
INFO 12/06/2005 22:58:06 Check mode - transferring validated config to remote node.  
INFO 12/06/2005 22:58:06 Configuration transferred successfully.  
INFO 12/06/2005 22:58:06 Calculated a check interval of 2.5 seconds.
```

If the check mode exits with a non-zero return code or output similar to the above then the configuration is valid – if there is a problem with any component in the specified application configuration a message should indicate the problem, allowing a speedy resolution.

Much more detailed information on the various Lems monitors which are typically used can be found in the section starting on page 111.

⁴Though it should be noted that this location can be overridden by changing the setting for the “logs” value in the Lems configuration file for an application if desired.

10 Starting the Cluster

Previously the quickest mention to start the cluster has been shown briefly. This section describes in more detail how the cluster daemon, (“cldaemon”) can be started and stopped to manage the cluster, from the point of view of the cluster administrator. For more information on the detailed technical implementation, please see the section beginning on page 190.

The cluster daemon, inter-changeably referred to as “cldaemon”, is the crux of the cluster environment, since it handles fail-over of applications (amongst other things) – thus it needs to be running before you are able to start any applications⁵.

The daemon can be started in two different ways; either directly on the command or via a high-level command known as “clform” (as shown previously).

This section describes the necessary steps of calling the “cldaemon” process directly to “form” the cluster. This is useful when the administrator is new to the product since it gives an insight into how exactly the cluster goes about the process of “forming a cluster”.

However once familiar with the workings of the cluster use of the simpler “clform” command is preferred. The “clform” utility is described following the use of “cldaemon” directly on the command line.

Typically the cluster would be formed when the nodes are booted – though if they are started at different times, then the cluster might start with only a single node in it. Details on automatic cluster formation on reboot and the options to consider are detailed on page 165. The current recommendation is that formation of the cluster should be a manual process rather than performed as part of the boot sequence of the servers.

The above is particularly important if applications are configuration to automatically start following the reboot. This is because if both nodes have failed at the same time the administrator should consider carefully on which nodes to form the cluster - ideally they should use the nodes last used for an application (if it was running prior to the reboot / machine failure).

10.1 Forming a Cluster using “cldaemon”

If the administrator has access to several windows, one particularly useful test is to run the cluster daemons in the “foreground”, at least for a little while. If the cluster is running it should be stopped before continuing with the commands here. To stop the cluster run:

```
# clhalt --force
```

Now to run the cluster in the foreground in the current window, use the following commands on both machines, starting the commands around the same time:

```
# cldaemon --form --verbose --file -
```

The above options indicate the cluster should be formed, with verbose logging, using standard output as the log file. Many more options are supported -, please see the documentation regarding the cluster daemon starting on page 190 for more information.

Since the above command will lock the terminal in question, please ensure another window is available on each node to run further commands.

⁵ This is not totally true - Linuxha.net does support starting applications without the cluster daemon for maintenance and disaster recovery handling.

This is the output from the primary node, (the first in the configuration file) whilst the cluster forms:

```
INFO 07/06/2005 06:24:21 Global section configuration validation complete
ERROR 07/06/2005 06:24:21 Unable to read from lock daemon.
INFO 07/06/2005 06:24:21 Locking daemon is not running - will attempt to start.
INFO 07/06/2005 06:24:23 Successfully started locking daemon.
INFO 07/06/2005 06:24:23 Network daemon is not running - will attempt to start.
INFO 07/06/2005 06:24:24 Successfully started network daemon.
INFO 07/06/2005 06:24:24 Started local server on port 9900
INFO 07/06/2005 06:24:25 Ping of 192.168.100.34 OK (network drbd)!
INFO 07/06/2005 06:24:25 Found IP for remote server: 192.168.100.34
INFO 07/06/2005 06:24:25 Waiting to form cluster with remote node.
INFO 07/06/2005 06:24:27 Response to ECHO was FORMING (our state=FORMING)
INFO 07/06/2005 06:24:27 Response to ECHO was STARTING (our state=FORMING)
INFO 07/06/2005 06:24:27 Response to ECHO was UP (our state=FORMING)
INFO 07/06/2005 06:24:27 Both nodes agree UP
ERROR 07/06/2005 06:24:29 Time difference between nodes is >10 minutes - require --
force to start!
```

The above output shows a common problem – the times used on the two nodes should be as close as possible - the above aborted the formation of the cluster because the time difference was too great - even though the cluster managed to form successfully.

After correcting the time the out will appear similar to the following:

```
INFO 06/06/2005 23:14:16 Added request ISVALID (Check Application valid nodes).
INFO 06/06/2005 23:14:16 Added request PNODE (Get application preferred node).
INFO 06/06/2005 23:14:16 Added request DEPENDS_ON (Get application dependencies).
INFO 06/06/2005 23:14:16 Added request AUTOSTARTLIST (Get list of auto-start
applications).
INFO 06/06/2005 23:14:16 Added request STOPPING_APP (Change Application state to
STOPPING).
INFO 06/06/2005 23:14:16 Added request RECONFIGURE (Rereconfigure running cluster
daemon).
WARN 06/06/2005 23:14:16 No validated build for samba - not registering with cluster.
INFO 06/06/2005 23:14:16 Validated Build for apache is valid - will register with
cluster.
INFO 06/06/2005 23:14:16 Application apache has been registered with the cluster
daemons
```

Please notice that the application that was created previously has been registered with the cluster - this means that a valid, checked configuration file for that application was found and that the application would be recognised as valid by the cluster.

In the above configuration a build an a “samba” application is under way – though the cluster does not register it yet since it does not match a valid checksum for the application.

At this point the cluster daemon processes are running, and notice that they have also started the lock daemons and network daemons on each node as well. The main cluster daemon will continue to do so until killed off. Since they are running attached to terminals terminate these sessions and instead run:

```
# cldaemon --form --verbose --detach
```

This will use the default log file to generate output to and run the program as a true daemon. You can check to see if the cluster is running by issuing a command such as:

```
# ps -ef | grep clustername
```

You would expect to see the following on both nodes if the cluster is running:

```
root      3496      1  0 23:12 ?          00:00:00 cllockd-s14cluster
root      3498      1  0 23:12 ?          00:00:00 clnetd-s14cluster
root      3506      1  2 23:16 ?          00:00:00 cldaemon-s14cluster
```

The processes started have their names changed to include the cluster name (as configured in the “name” element in the “global” section the clconf.xml file). This can then be used by the administrator to monitor/kill Linuxha.net system processes if so desired. However the

administrator should be aware that Linuxha.net provides utilities for direct management of all daemon processes without the administrator requiring to know the relevant process ID's.

10.1.1 When to use force to form a cluster

The "--force" option performs two different actions, but as the name indicates the purpose is to ensure that a cluster is formed, even if by default perhaps it should not be. The two purposes of using "--force" when attempting to start the cluster are:

- To overcome a too large a time difference. If the difference between the system times on both machines is over 10 minutes out then the cluster will abort with a warning. This is because accurate time is necessary since application start times are recorded and used during the process of application start-up under certain operating conditions.
- To form the cluster both machines need to agree on the status – a certain period of time is allotted before giving up on the other server as being dead. For reasons of data sanity the cluster does not form in such cases unless forced to.

10.2 How a Cluster is Joined

If a node has died or the cluster was started without it (using the "--force" option), then it is possible for that node to become part of the running cluster - this is known as "joining" the cluster. To attempt to get the "cldaemon" on the current machine to attempt to join an already running cluster on the other server, use the following command:

```
# cldaemon --join --detach --verbose
```

This might produce something similar to the following in the log file in this instance:

```
INFO 06/06/2005 23:18:56 Global section configuration validation complete
INFO 06/06/2005 23:18:56 Locking daemon already appears to be running.
INFO 06/06/2005 23:18:56 Network daemon already appears to be running.
INFO 06/06/2005 23:18:56 Started local server on port 9900
INFO 06/06/2005 23:18:56 Ping of 192.168.100.34 OK (network drbd)!
INFO 06/06/2005 23:18:56 Found IP for remote server: 192.168.100.34
INFO 06/06/2005 23:18:56 Waiting to join cluster with remote node.
INFO 06/06/2005 23:18:58 Registering new application: apache
INFO 06/06/2005 23:18:58 resp2=DOWN,...,0,1,s14s1+s14s2
INFO 06/06/2005 23:18:58 Setting valid nodes: s14s1,s14s2
INFO 06/06/2005 23:18:58 Added request STOP_NBD (Stop DRBD resources).
INFO 06/06/2005 23:18:58 Added request START_NBD (Start DRBD resources).
. . .
INFO 06/06/2005 23:18:58 Added request GETVALIDNODES (Get Application valid nodes).
INFO 06/06/2005 23:18:58 Added request SETVALIDNODES (Set Application valid nodes).
INFO 06/06/2005 23:18:58 Added request ISVALID (Check Application valid nodes).
INFO 06/06/2005 23:18:58 Added request PNODE (Get application preferred node).
INFO 06/06/2005 23:18:58 Added request DEPENDS_ON (Get application dependencies).
INFO 06/06/2005 23:18:58 Added request AUTOSTARTLIST (Get list of auto-start
applications).
INFO 06/06/2005 23:18:58 Added request STOPPING_APP (Change Application state to
STOPPING).
INFO 06/06/2005 23:18:58 Added request RECONFIGURE (Rereconfigure running cluster
daemon).
INFO 06/06/2005 23:18:58 Node s14s1 has now joined the cluster
```

Joining the cluster has no impact to the applications currently running in the cluster – they will continue to run. Any further applications you wish to start can now of course be started on either node. If data replication resulted in Stale data (since the node in question actually crashed), then joining the cluster will allow the data replication for any running application to be started shortly.

10.3 Forming a Cluster using "clform"

Although calling the "cldaemon" directly is a supported method of forming (or joining) the cluster, in practise it can become tiresome. To ensure the process of cluster formation is as straightforward as possible the "clform" utility is provided.

This command offers a limited set of command line options, but provides enough functionality to form the cluster under most circumstances. There are still occasions however when the

"cldaemon" commands described previously must be resorted to due to the fine-grain control of the cluster formation process that is available.

By default the utility does not attempt to force the cluster and of course uses the default time out when forming a cluster. Both daemons are run detached in verbose logging mode. Once the cluster forms it will return to the command line with a status of 0.

If an error occurs a non-zero returned code is specified and where possible an error message will be shown to the standard error device. If no error does appear than check the log file for the cluster. As previously mentioned the log file will have the following name:

```
/var/log/cluster/cldaemon-clustername.log
```

The "clform" command is explained in more detail, along with supported command line options, starting on page 141.

It should be noted now that "clform" is also able to start cluster applications automatically as part of the process. This does require additional attributes to be defined for the application, and can be particularly useful in more complex configurations.

Finally to actually form the cluster using the "clform" command, run the following on either one of the nodes:

```
# clform --noapps
```

In this instance the "--noapps" argument has been given to ensure that no applications are started. The output generated is much more simplified compared to using "cldaemon" directly:

```
INFO 06/06/2005 23:39:00 Validated checksum for cluster configuration
INFO 06/06/2005 23:39:00 SSH communication to s14s2 will be:
INFO 06/06/2005 23:39:00 192.168.100.34 ("drbd" network)
INFO 06/06/2005 23:39:00 Checking that the cluster is not already running...
INFO 06/06/2005 23:39:00 *** ATTEMPTING TO FORM CLUSTER s14cluster ***
INFO 06/06/2005 23:39:00 Starting cldaemon on s14s1...
INFO 06/06/2005 23:39:01 Starting cldaemon on s14s2...
INFO 06/06/2005 23:39:02 Waiting for cluster to form...
INFO 06/06/2005 23:39:06 Cluster s14cluster started successfully.
```

10.4 Forming a Cluster on machine boot

Before describing the steps necessary to form a cluster on machine boot there are several important questions that require consideration;

- Should the cluster form if only one machine is available at the end of the time-out interval?
- Should the formation of the cluster occur as a foreground task or background "init" task?
- If the other node is already running the cluster services should the new node automatically join the existing cluster?
- Should auto-start applications be automatically started?
- Should the default time-out specified in the cluster configuration be used?

If cluster formation is required at machine start-up then Linuxha.net provides the following command to build a facility tailored for the start-up of the cluster at machine boot time:

```
/sbin/cluster/tools/clstartup
```

To make use of this facility a configuration must be present otherwise calling the command will fail. The configuration file necessary has the following path and must be present on both nodes:

```
/etc/cluster/clstartup.xml
```

A typical example of this XML file would look similar to the following:

```
<?xml version="1.0" standalone="yes"?>
<clstartup_config>
  <allow_single_node>yes</allow_single_node>
  <background_startup>no</background_startup>
  <join_existing_cluster>no</join_existing_cluster>
  <autostart_apps>yes</autostart_apps>
  <timeout>120</timeout>
</clstartup_config>
```

The five settings correspond to the questions mentioned on the previous page. The table below describes the options in a little more detail.

Setting	Purpose
allow_single_node	Should be set to "yes" or "no". <ul style="list-style-type: none"> ➤ <i>yes</i> - if after the time-out period the node in question is not able to communicate with the other node a single node cluster is formed. ➤ <i>no</i> - the cluster formation fails if both nodes are not present to create a cluster when the time-out period expires.
background_startup	Should be set to "yes" or "no". <ul style="list-style-type: none"> ➤ <i>yes</i> - the formation of the cluster is handled as a background task allowing the rest of the run-control scripts to continue processing. ➤ <i>no</i> - foreground the task - all remaining run-control scripts wait for the cluster to form, or the time-out period to expire⁶.
join_existing_cluster	Should be set to "yes" or "no". <ul style="list-style-type: none"> ➤ <i>yes</i> - if the cluster is already formed and this command is run then this node attempts to join it. ➤ <i>no</i> - if the cluster is already formed then do not attempt to join it.
autostart_apps	Should be set to "yes" or "no". <ul style="list-style-type: none"> ➤ <i>yes</i> - if the cluster is successfully formed any application that has an "autostart" attribute set to "yes" is started in the cluster - along with any dependencies if required. ➤ <i>no</i> - no applications are automatically started.
timeout	If present this setting is given in seconds and overrides that defined in the cluster configuration setting "timings.clusterform", (which if not present defaults to 300).

Since the background start-up is not yet supported as of version 1.0.0 it is recommended that if added to run-control scripts it is added after the "getty" process is started (otherwise console log could be delayed by many minutes).

The above point is particularly important since if there is a problem with the configuration it will allow the administrator to log in rather than waiting for the time-out period to expire.

⁶ In the current version 1.0.0 this setting is currently ignored - all processing occurs in the background.

11 Managing Applications in the Cluster

11.1 Starting Applications with "clstartapp"

Once the cluster daemons have been started the next step is to start the application (or applications) in the cluster. Each application must be started separately by running the "clstartapp" command. This command takes various options but typically just needs the name of the application to start:

```
# clstartapp --application apache
```

The above command would silently start the "apache" application on the current node. The command should be run on the node on which the application is to be executed on – if this particular node is not suitable (for various reasons described later) the start-up of the application will fail. Later a more straightforward command will be described for handling applications that takes more account of the current cluster state.

A more common start-up command might be something similar to the following:

```
# clstartapp --application apache --verbose --file -
```

In the above example the "--file" option has been used with "-" to indicate that the default location for sending progress messages to when starting the application in verbose mode should be over-ridden and instead make use of Standard Output.

Running the above commands assumes the cluster is running in a particular state:

- [1] The cluster daemons are running.
- [2] A valid copy of the data is available locally, or via the DRBD network to the remote system.
- [3] The current node is actually running as part of the cluster.
- [4] The specified application is configured and built correctly, but no currently running in the cluster.

If all the above are true, then the clustered application should be able to start, and since the command has been run in verbose mode, output similar to the following should be produced:

```
INFO 12/06/2005 23:03:22 Validated checksum for cluster configuration
INFO 12/06/2005 23:03:22 Checked that node names resolve to IP addresses
INFO 12/06/2005 23:03:22 Validated Build run has completed against this
configuration.
INFO 12/06/2005 23:03:22 Maximum start-up time for application: 20
INFO 12/06/2005 23:03:22 drbd kernel module loaded already on sl4s1
INFO 12/06/2005 23:03:22 drbd kernel module loaded already on 192.168.100.34
INFO 12/06/2005 23:03:23 Local DRBD devices started successfully.
INFO 12/06/2005 23:03:23 Ssh communication to sl4s2 via 192.168.100.34.
INFO 12/06/2005 23:03:23 DRBD: Skipping ENBD decisioning and relying on meta data...
INFO 12/06/2005 23:03:23 Attempting to start DRBD services on sl4s2.
INFO 12/06/2005 23:03:24 DRBD devices started successfully on sl4s2.
INFO 12/06/2005 23:03:24 Validated consistency of available data for DRBD.
INFO 12/06/2005 23:03:24 Both data copies believed good.
INFO 12/06/2005 23:03:24 Locking will be attempted via port 9849
INFO 12/06/2005 23:03:24 Successfully connected to lock server.
INFO 12/06/2005 23:03:24 Attempting to register application apache as starting...
INFO 12/06/2005 23:03:24 Application registered successfully as starting.
INFO 12/06/2005 23:03:24 Checking for existing primary application IP addresses...
INFO 12/06/2005 23:03:26 No application primary IP addresses found.
INFO 12/06/2005 23:03:26 Attempt to get lock for NBD_CLIENT
INFO 12/06/2005 23:03:26 Attempting to make local DRBD devices primary...
INFO 12/06/2005 23:03:27 Attempt to release lock for NBD_CLIENT
INFO 12/06/2005 23:03:27 All local DRBD now primary.
INFO 12/06/2005 23:03:27 Running "/sbin/fsck -t ext3 -a /dev/drbd0"...
INFO 12/06/2005 23:03:27 Running "PATH=$PATH:/sbin:/bin:/usr/sbin; mount -t ext3 -o
rw /dev/drbd0 /apache"...
INFO 12/06/2005 23:03:27 File systems mounted on DRBD devices.
INFO 12/06/2005 23:03:27 Attempt to get lock for NET
INFO 12/06/2005 23:03:27 Configuring 192.168.0.102: ifconfig eth0:1 inet
192.168.0.102
```

```
INFO 12/06/2005 23:03:27 Sending Builtin Gratuitous arp for eth0:1
INFO 12/06/2005 23:03:27 Configuring 192.168.0.103: ifconfig eth0:2 inet
192.168.0.103
INFO 12/06/2005 23:03:27 Sending Builtin Gratuitous arp for eth0:2
INFO 12/06/2005 23:03:27 Attempt to release lock for NET
INFO 12/06/2005 23:03:30 Applications start completed successfully
Application apache started successfully
```

Of course if the application is already running in the cluster a suitable error message will be produced, as the following example shows:

```
INFO 12/06/2005 23:09:05 Validated checksum for cluster configuration
INFO 12/06/2005 23:09:05 Checked that node names resolve to IP addresses
INFO 12/06/2005 23:09:05 Validated Build run has completed against this
configuration.
INFO 12/06/2005 23:09:05 Maximum start-up time for application: 20
INFO 12/06/2005 23:09:05 drbd kernel module loaded already on sl4s1
INFO 12/06/2005 23:09:06 drbd kernel module loaded already on 192.168.100.34
INFO 12/06/2005 23:09:06 Local DRBD devices started successfully.
INFO 12/06/2005 23:09:06 Ssh communication to sl4s2 via 192.168.100.34.
INFO 12/06/2005 23:09:06 DRBD: Skipping ENBD decisioning and relying on meta data...
INFO 12/06/2005 23:09:06 Attempting to start DRBD services on sl4s2.
INFO 12/06/2005 23:09:07 DRBD devices started successfully on sl4s2.
INFO 12/06/2005 23:09:07 Validated consistency of available data for DRBD.
INFO 12/06/2005 23:09:07 Both data copies believed good.
INFO 12/06/2005 23:09:07 Locking will be attempted via port 9849
INFO 12/06/2005 23:09:07 Successfully connected to lock server.
INFO 12/06/2005 23:09:07 Attempting to register application apache as starting...
ERROR 12/06/2005 23:09:07 Application apache is already running!
```

The log messages shown previously are only relevant for a cluster running in an optimal condition. The more interesting/important output occurs when the cluster is **not** running in such a manner (for example when both nodes are not available, or certain network connectivity is not available). For more information on these considerations see the technical information starting on page 165.

Please note that the default file for the output if running “clstartapp” in verbose mode is the following:

```
/var/log/cluster/clstart.application.log
```

The start script specified to run the programs/services for this particular application will log output to the following destination, (unless the script redirects output and errors elsewhere):

```
/var/log/cluster/application.start.log
```

Note that the time taken to start an application can vary greatly depending on the performance of the nodes, the status of the cluster, as well as the number of file systems required to mount, their size and type.

11.2 Some typical Error Conditions when Starting Applications

As indicate a later section in this document can be referred to for detailed implementation information regarding the “clstartapp” command. However this is really aimed at developers, and probably contains more information that system administrator care to know. Hence this section describes some of the more common failure conditions that might occur when starting the application. For other failure conditions please see the documentation starting on page .

1. Software indicates an incorrect Checksum has been found
If the application has previously been built correctly then this error indicates that the configuration file has been changed, and the “clbuildapp” stages re-run. Details on how, why and the effects of running “clbuildapp” when a application already exists can be found later in this document. The alternative approach, to be used with extreme caution, is to add the “--nochecksums” option to the command line to turn off configuration file validity checking.

2. Specified application is not registered with the cluster

This occurs because the application specified either does not exist, or more likely it has not yet been successfully built. In such cases the error might be indicated with a message similar to the following:

```
ERROR 30/01/2004 22:32:22 Could not find appconf.xml for fred
```

In this case it is recommended that the application is built correctly using the “clbuildapp” before continuing.

1. Application exists, but indicates “--build” stage needs running.

If an error message appears similar to the one below it indicates that either as it states the “--build” stage for the application has not been completed using the “clbuildapp” utility, or the configuration file for the application has been altered, but the build stage has not been re-run. This is necessary to ensure the configuration is valid.

```
ERROR 30/01/2004 22:37:27
ERROR 30/01/2004 22:37:27 The configuration file for this application has been altered
ERROR 30/01/2004 22:37:27 but the --build stage has not yet been rerun. Do that first
ERROR 30/01/2004 22:37:27 before running with the --sync option.
```

1. Application will not start on node

If this occurs and the previous two conditions are not true then it indicates that the node is probably not running a cluster daemon. This is typically indicated by the following lines appearing in the log file, or to the terminal:

```
WARN 31/01/2004 18:15:51 Unable to register application (not response)
ERROR 31/01/2004 18:15:51 Unable to contact local cluster daemon.
ERROR 31/01/2004 18:15:51 Will not start the application due to the risk of data
corruption.
```

If necessary it is possible to start an application on a node without a cluster daemon running – *though this is not recommended*. To proceed with a startup with the daemon simply add the “--force” option to the command line.

2. Application does not start, even when the “--force” option is used.

This will occur because of either of the following reasons; either one of the “primary” application IP addresses is not unique; or the specified application is running elsewhere using this IP address. In both cases you will see the following message in the log file:

```
ERROR 31/01/2004 21:58:39 It appears that the IP address for this application is in
use.
ERROR 31/01/2004 21:58:39 Either the application is running already, or the configured
ERROR 31/01/2004 21:58:39 IP address is not suitable.
```

3. Software indicates an unknown state

It is still theoretically possible that the cluster is in such a state that the software will refuse to start the application. In such cases the recommended resolution is to perform the “--sync” option of the “clbuildapp” utility from the node that you know has the copy of data that is to be used. After this has been completed attempt to start the application again.

4. DRBD Device Errors

It is possible that the DRBD devices on the local machine where the application is to be started will not start. (If the remote ones fail to start the application will start-up with stale copies).

In these cases the problem is likely to be caused by one or more of the following conditions;

- **Missing Device Files**
The required entries in /dev do not exist currently. This is particularly possible if further devices have been added to cluster and the rebuild has been completed correctly.
- **Missing Kernel Module**
This might occur if the server kernel has been upgraded and the DRBD module has

- not been recompiled.
- Mismatch in maximum devices supported
By default DRBD is configured to manage just two devices unless options are specified during the module load. Hence it is possible that if the module has been loaded manually that this number is lower than those device numbers configured by Linuxha.net.

In this case the module should be unloaded with “rmmod” and the application start-up command tried again.

It is often useful to check the status of DRBD on a local host. This is done simply by:

```
# cat /proc/drbd
```

5. The cluster configuration file does not exist
An error message of this nature occurs if you are attempting to start the application when no actual cluster configuration seems present. The most likely cause of this problem is the removal of the “/etc/cluster/clconf.xml” file which is required for the software to function.
6. Invalid permissions or status of “clconf.xml” file
For security reasons explained previously (the presence of the communication key), the “/etc/cluster/clconf.xml” must have permissions “rw-----” (read/write by owner only), and must be owned by “root”. If this is not the case the application will not try to make use of this configuration file.
7. Invalid XML configuration file
If the XML parser is unable to read the configuration file for the application then the attempt to start the application can not take place. Essential information for those new to XML can be found starting on page .
8. Missing “.clbuild.md5” Configuration file
This file is generated when an application is built. Such files should not be removed from the application configuration directory. If this file is missing for an application the “clbuildapp” command must be re-run to regenerate it. The files are used to validate that the application has been successfully configured and thus act as a further check to ensure a invalid configuration is not made use of.
9. Configuration file for later software revision
The error typically occurs if the application believes that the version of the file is too new for the installed version of the Linuxha.net software. Such errors can be overcome by checking the “version” details of the application configuration, or upgrading to the latest required revision of software.
10. Missing nodes or invalid nodes in cluster configuration file
This can only occur if the cluster configuration files and “.clbuild” files have been copied from a currently running cluster to a machine that was not named as part of that cluster. Please ensure that all cluster configuration steps using “clbuild” are repeated to ensure the current cluster and application configurations are revalidated.
11. Status directory problems
As mentioned in previous sections the directory “/etc/cluster” contains sub-directories that include resource and status information regarding the cluster. This directory should not be modified or removed apart from by the tools provided as part of Linuxha.net. Errors of this nature are typically caused by manipulation or removal of files or directories, without which the cluster software can not function correctly.
12. Errors indicated during IP-level checking

Sometimes it is possible to see errors similar to the following appear when trying to start an application:

```
clstartapp when no interfaces for prod interface available (if running with ip checking)
```

In such cases this occurs because the details in the IP check-list for the application are not meeting the required pass percentage. If the IP addresses are valid, and the “ping” command appears to work with them, consider the protocol specified for the “ping”. It is likely to be UDP or TCP and in such cases you hosts may need explicit configuration in the “/etc/inetd.conf” file or “/etc/xinetd.d/echo-udp” for example to ensure “echo” requests for this protocol are actually enabled.

11.3 Checking Application Status

Assuming that any problems, (if there were any!) have been overcome the administrator next will probably wish to validate the status of the cluster currently. Most cluster status checking can be carried out using the “clstat” command. You can run it without arguments to get a summary status:

```
# clstat
```

In this case the output will appear similar to the following:

```
Cluster: sl4cluster - UP
```

Node	Status
sl4s1	UP
sl4s2	UP

Application	Node	State	Started	Monitor	Stale	Fail-over?
apache	sl4s1	STARTED	0:00:16	Running	0	Yes
samba	N/A	DOWN	N/A	N/A	N/A	Yes

The output should mostly be self-explanatory - the exceptions being “Monitor”, “Stale” and “Fail-over?”. These three fields are very important to the availability of the application and must be understood.

- *Monitor* - This indicates whether a Lems daemon is running for the application and will either be “Yes” if one is running otherwise “No” if not. If the application is not running that this will show “N/A” instead, as indeed it will during application state transitions⁷.
- *Stale* - This indicates how many file systems are currently considered “stale” - that is the data copies are not fully synchronised on the two nodes. Under normal conditions with will be set to “0” otherwise if stale data does exist, it indicates the number of file systems affected.
- *Fail-over?* - This indicates whether or not the software is configured to fail-over to the other node if it is available.

The standard output generated by “clstat” gives a summary of the status of the cluster - more information is available to the administrator via additional command line arguments, the most common being “--application” to show more detail for the specified application:

```
# clstat --application apache
```

⁷ A state-transition will occur when the application changes from STARTING to STARTED or from STOPPING to STOPPED. This message may also appear for a short time when the cluster daemon is re-started (due to software failure) on a node already running the application.

The output generated will appear similar to the following:

```
Cluster: sl4cluster - UP
```

Application	Node	State	Runnig	Monitor	Stale	Fail-over?
apache	sl4s1	STARTED	0:00:19	Running	0	Yes

File Systems

Mount Point	Valid	Type	State	% Complete	Completion
/apache	both	drbd	Sync		

Process Monitors

Name	Status	Restarts	Current	Reset at
httpd	Running	3	0	N/A

General Monitors

Type	Name	Status
Flag Check	flag_check	Running
FS Monitor	fsmonitor	Running
IP Monitor	ipcheck	Running

This output show details of the monitors and details of each file system that forms the particular application.

The "IP Assignment" module is meant to be stopped under most circumstances since it is responsible for moving the IP address for the application!

Each sub-section of the output generated will now be described separately. Again a full understanding of this output is necessary if you are to manage the cluster software correctly.

11.3.1 The "File Systems" information

This section provides detailed information on the status of each file system that this particular application includes. This information is virtually important since it indicates how available the data is. Each column is now explained.

- *Mount Point* - The mount point of a particular file system. Previous versions of the software displayed the underlying volume group and logical volume which was found to be less useful for most administrators.
- *Valid* - Which copy of the data is currently valid. This will be either be "both", "local" or "remote". Local and remote are based on the location of the application currently, "local" obviously being the node on which is is currently running. Of course in most cases this should be "both"!
- *Type* - For current versions of Linuxha.net this is always "drbd" - older versions would indicate "fs1" or "raid1".
- *State* - The synchronisation state of the local volume - can be "sync", "syncing" or worst of all "unsync".
- *% Complete* - If the state is currently "syncing" then this is the current % complete of the operation to synchronise the mirror copy is.
- *Completion* - An estimated duration from now when the synchronisation operation will complete, (given in days, hours, minutes and seconds).

The "% Complete" and "Completion" are both estimates and will vary second to second based on current operating conditions of both nodes, being particularly affected by other disk utilisation.

If data synchronisation is required for an application then please be aware that all file systems for an application will synchronise simultaneously. Later versions of Linuxha.net will support user-defined prioritisation of this re-synchronisation - particularly useful in certain environments.

11.3.2 The "Process Monitors" information

As shown previously if the user wishes to monitor the status of an application then typically one or more process monitor entries will exist for an application. Each monitor will be given a separate series of details under the following headings:

- *Name* - The name given to the process monitor in the Lems configuration file - user definable, though should be a alphanumeric value without white space.
- *Status* - This will be either "running" or "stopped" indicating whether the monitor is currently checking the processes or dormant.
- *Restarts* - The maximum number of restarts of the application that are configured into the monitor before it indicates a fail-over to the other node should be attempted.
- *Current* - The current number of attempted fail-overs.
- *Reset at* - The "Current" number of attempted fail-overs is reset after a time-period - this is the time at which the "current" setting will be automatically cleared back to zero.

11.3.3 The "General Monitors" information

There are several other monitors which take care of flags and network state functionality for the application. Typically these are not managed by the user - for detailed information on the purpose of these highly important monitors see the Lems technical information starting on page .

The table below gives summary information regarding monitors typically used however:

Monitor	Purpose
Flag Check	Checks for the existence of certain files and if found deactivates specified monitors, (good for maintenance scripts).
FS Monitor	Monitor the synchronisation status of the file systems and re-synchronises file systems when necessary.
IP Monitor	Checks for the availability of the networking infrastructure by the use of ICMP ping requests to list of hosts.

11.4 Stopping Applications

At some point it will obviously be necessary to take down an application. The "clhaltapp" utility is provided for this purpose. This command should be run on the server which is currently hosting the application.

It is typically run with the following format:

```
# clhaltapp --application application --verbose
```

The output this generates in the log file would be similar to the following:

```
INFO 12/06/2005 23:39:48 Validated checksum for cluster configuration
INFO 12/06/2005 23:39:48 Checked that node names resolve to IP addresses
INFO 12/06/2005 23:39:48 Validated Build run has completed against this
configuration.
INFO 12/06/2005 23:39:48 Maximum shutdown time for application: 10
INFO 12/06/2005 23:39:48 Locking will be attempted via port 9849
INFO 12/06/2005 23:39:48 Successfully connected to lock server.
WARN 12/06/2005 23:39:48 Ssh communication to sl4s2 via 192.168.100.34.
INFO 12/06/2005 23:39:48 Locking will be attempted via port 9849
INFO 12/06/2005 23:39:48 Successfully connected to lock server.
INFO 12/06/2005 23:39:48 Lems Daemon aborted.
INFO 12/06/2005 23:39:48 Applications stop completed successfully
INFO 12/06/2005 23:39:48 Attempting un-mount of /apache...
INFO 12/06/2005 23:39:48 unmount failed with: 1
INFO 12/06/2005 23:39:50 unmount failed with: 1
INFO 12/06/2005 23:39:51 File system /apache un-mounted.
INFO 12/06/2005 23:39:51 Attempt to get lock for NET
INFO 12/06/2005 23:39:51 Removing 192.168.0.102: ifconfig eth0:1 down
INFO 12/06/2005 23:39:51 Removing 192.168.0.103: ifconfig eth0:2 down
INFO 12/06/2005 23:39:51 Attempt to release lock for NET
INFO 12/06/2005 23:39:51 Attempt to get lock for NBD_CLIENT
INFO 12/06/2005 23:39:51 Stopping all active DRBD connections on sl4s1.
INFO 12/06/2005 23:39:51 Attempt to release lock for NBD_CLIENT
INFO 12/06/2005 23:39:51 All active DRBD devices stopped locally.
INFO 12/06/2005 23:39:51 Attempt to get lock for NBD_SERVER
INFO 12/06/2005 23:39:51 Stopping all active DRBD devices on sl4s2.
INFO 12/06/2005 23:39:53 Attempt to release lock for NBD_SERVER
INFO 12/06/2005 23:39:53 All active DRBD devices stopped remotely.
Application apache shutdown successfully.
```

There are other command line options available which can be used for various purposes - see the detailed technical documentation regarding starting and stopping applications beginning on page 165.

In the above output noticed that the key part of shutting down the application quickly is ensuring the "stop" script (if any has been specified - and it should be), quickly stops the application. Even if this script fails to stop the application it will not cause the shut down of the application to abort - though a failure will be noted.

The default location for the output when running in verbose mode is:

```
/var/log/cluster/clhalt.application.log
```

For the application stop script the standard output and error are redirected to the following files respectively:

```
/var/log/cluster/application.stop.log
```

and;

```
/var/log/cluster/application.stop.errlog
```

11.5 Starting Applications (the easy way)

The “clstartapp” application is the lowest level of functionality that can be used to run an application under Linuxha.net. However for large-scale deployments consisting of many applications it does have some limitations, and hence the reason for the inclusion of the “clrunapp” utility.

This command is essentially a wrapper for the “clstartapp” as shown previously. However it provides a limited amount of intelligence to handle applications, which make it the best tool to use in most cases. To use this utility simply run:

```
# clrunapp --application apache
```

This will show output similar to the following:

```
INFO 12/06/2005 23:43:24 Validated cluster configuration.
INFO 12/06/2005 23:43:24 Validated application configuration.
INFO 12/06/2005 23:43:24 Successfully connected to cluster sl4cluster.
INFO 12/06/2005 23:43:24 Verified that application apache is registered.
INFO 12/06/2005 23:43:24 Current application state : DOWN
INFO 12/06/2005 23:43:24 Application apache depends on: <NONE>
INFO 12/06/2005 23:43:24 Application apache will be started on node sl4s1
INFO 12/06/2005 23:43:24 Starting apache using command:
INFO 12/06/2005 23:43:24 /sbin/cluster/clstartapp --application apache --maxdelay 30
--verbose
INFO 12/06/2005 23:43:31 Application apache started after 7 seconds.
```

This tool checks the dependency information (if present) for the application and will only allow the application to be started if all dependencies are met, (by default). There are some other command line options available, (described on page 140), but typically these are not needed.

For a simple cluster configuration this will do nothing different from the “clstartapp” example shown previously. It will start the application using verbose logging and return to the command line once complete.

The major difference with this command compared with “clstartapp” is that it is node independent – it might start the specified application locally or on the remote node depending on the application characteristics and the state of the cluster generally.

To take full advantage of this command some additional attributes must be specified as part of the application configuration file.

Information on the more advanced attributes from the application configuration files can be found on page 140.

11.6 Managing application Monitoring

One of the most common things activities that is undertaken whilst an application is running in a cluster is to stop the monitoring of the application for a short period whilst non-standard conditions are expected to prevail.

For example the administrator might wish to upgrade the “httpd” binary or modules, but does not wish to fully stop the application; what is required is for the Lems system to ignore the process monitor functionality for a short duration.

Fortunately such functionality is readily built in - notice that the output shown for the application status on page 81 includes a “flag” module. With this module the administrator is able to control whether other modules in Lems are active or not - even the “flag” module itself!

The flags module works as follows - note the name of the Lems module that monitoring is to be suspended for, (under the Name heading for the process and general monitors). Then simply create a file called the name of the monitor in the following directory:

```
/etc/cluster/application/flags
```

Using the example “apache” application, if you wish to stop the process monitor “httpd” then the following command could be used:

```
# touch /etc/cluster/apache/flags/httpd
```

This needs to be performed on the node which is currently running the application!

Shortly after touching the file please notice that the following message in the “Lems” log file for the application will appear:

```
INFO 26/11/2003 22:25:23 Monitor flag_check returned: 1 => [STOP httpd;]
INFO 26/11/2003 22:25:23 Check httpd has been stopped, (RC=1)
```

This indicates that the flag has been noticed and the “flag_check” module told “Lems” to stop monitoring “httpd”. This can be confirmed by checking the detailed status of the application.

```
# clstat --application apache
```

Notice that the “Process Monitors” section of the output now looks like:

```
Process Monitors
      Name      Status  Restarts   Current      Reset at
      httpd     Stopped    3         0           N/A
```

If you wish you can be now remove the flag file:

```
# rm /etc/cluster/apache/flags/httpd
```

Now the “Lems” log will record the fact after a short delay:

```
INFO 26/11/2003 22:29:28 Monitor flag_check returned: 1 => [START httpd;]
```

Now the “Process Monitors” section of the application status will indicate that the monitor is indeed running:

```
Process Monitors
      Name      Status  Restarts   Current      Reset at
      httpd     Running    3         0           N/A
```

It should be noted that the “Lems” monitor actually does listen on a socket and will respond to certain messages – a tool called “lemsctl” is available to interface to the daemon using this

communication channel. Information on available uses of this utility can be found in the Administration section of the document starting on page 174.

12 Application Removal

Since the life time of the cluster might be longer than one or more of the applications that have been defined for it, it is obviously necessary to provide a utility which can be used to remove the application from the cluster - this facility is provided by the "clremoveapp" tool.

The first point to remember is that the application must not obviously be running! It will actually check to ensure that this is the case - but the "--force" option can be used to override this!

It is also worth noting that the removal of the application from the cluster does not destroy the data - the volume group configuration information on both machines will be left untouched - allowing the administrator to add the application into the cluster later if necessary.

To remove the "apache" application from the cluster simply run the following command would be used:

```
# clremoveapp --application apache --verbose
```

If you do attempt to remove the application whilst it is still running you will get an error similar to the following:

```
INFO 12/06/2005 23:57:55 Global section configuration validation complete
ERROR 12/06/2005 23:57:55 Daemon responded with status STARTED to application status request!
```

The force option should only be used when as a last possible resort. Using this on a running application may result in data loss.

The utility supports a very limited set of command line arguments - these being described below - importantly included one to back up the configuration of the cluster prior to any changes taking place:

Flag	Purpose
--application X	The name of the application to remove from the cluster.
--verbose	The utility will report informational messages as it goes about whatever business it needs to. All lines will start with "INFO" and will appear on the standard output device.
--nobackup	Do not back up the contents of the configuration file - simply remove it (on both hosts).
--backupdir F	The directory to use to back up the configuration - must be specified unless the previous option has been specified.
--file X	This is used to specify the location which is used to store the verbose output - it can be specified as "-" to indicate the current terminal / Standard Output.
--force	Force the removal of the application - this is needed if the cluster daemons are not currently running.
--nochecksums	Allow the removal of the application even if the configuration file for the application does not match the validated checksums.

Once this utility has been run the specified application is no longer registered with the cluster. Prior to removing the application the following series of checks are carried out to ensure it is actually sensible to remove the application.

1. Check that the application does not appear to be running - this is only possible if the cluster daemons are actually running at this time.

2. Check that all file systems specified for the application are unmounted - on both nodes.
3. Check that both nodes agree that the specified application is defined - though do not need to check that they agree on what resources are used.

Point 2 above is important - since it must confirm the status of the application on both nodes, an application can only be removed once both nodes are available. Once all the specified checks have been performed then the following series of actions are carried out to remove the definition of the application:

- If the Cluster daemon is running then send it a message to ensure the application is removed from the registered series of applications.
- Any file of the format "NN.<application>.vg.lv" in the local directory "/etc/cluster/.resources/drbd" should be removed and replaced with a file called "NN.free", (so the device can be re-used by other applications).
- The above test/action should be repeated on the remote node.
- Any file of the format "NN.<application>.vg.lv" in the local directory "/etc/cluster/.resources/ports" should be removed and replaced with a file called "NN.free", (so the port can be re-used by other applications).
- The above test/action should be repeated on the remote node.
- If the file "/etc/cluster/.resources/fsmap/<application>" exists locally it should be removed.
- The above test/action should be repeated on the remote node.
- If the directory "/etc/cluster/<application>" exists locally then it should be removed, along with any files and sub-directories.
- The above test/action should be repeated on the remote node.

Once these steps have been carried out then the specified application will no longer exist as far as the cluster is concerned. If the cluster is currently running then the application will automatically be unregistered and will no longer appear in the output of the "clstat" utility.

When this command is carried out all of the files that defined the application, (such as the "/etc/cluster/<application>/appconf.xml") are archived by default. This is advisable - to turn this feature off specify "--nobackup" on the command line.

Currently none of the default log files that this package used on either host are currently removed - this must be performed manually by the administrator.

The output when an application is removed will appear similar to the following:

```
# clremoveapp -A samba -V
INFO 13/06/2005 00:05:00 Global section configuration validation complete
INFO 13/06/2005 00:05:00 Confirmed samba is not currently running.
INFO 13/06/2005 00:05:00 Will use IP address 192.168.100.34 for communication to
s14s2.
INFO 13/06/2005 00:05:00 Validated existance of /sbin/cluster/utlils/rmdir on node
s14s1
INFO 13/06/2005 00:05:00 Validated existance of /sbin/cluster/utlils/rmdir on node
s14s2
INFO 13/06/2005 00:05:00 Validated existance of /sbin/cluster/utlils/mkdir on node
s14s1
INFO 13/06/2005 00:05:01 Validated existance of /sbin/cluster/utlils/mkdir on node
s14s2
INFO 13/06/2005 00:05:01 Application unregistered from cluster.
INFO 13/06/2005 00:05:01 Attempting to backup cluster configuration on node s14s1.
INFO 13/06/2005 00:05:01 Saved cluster configuration in 38243 bytes on node s14s1
INFO 13/06/2005 00:05:01 Attempting to backup cluster configuration on node s14s2.
INFO 13/06/2005 00:05:02 Saved cluster configuration in 11622 bytes on node s14s2
INFO 13/06/2005 00:05:02 Successfully removed /etc/cluster/.status/samba/TIME (file)
from node s14s1
INFO 13/06/2005 00:05:02 Successfully removed /etc/cluster/.status/samba/.
(directory) from node s14s1
INFO 13/06/2005 00:05:03 Successfully removed /etc/cluster/.status/samba/.
(directory) from node s14s2
INFO 13/06/2005 00:05:03 Successfully removed drbd resource
/etc/cluster/.resources/drbd/10.samba.sambavg.cfg (file) from node s14s1
INFO 13/06/2005 00:05:03 Successfully removed drbd resource
/etc/cluster/.resources/drbd/11.samba.sambavg.logs (file) from node s14s1
```

```

INFO 13/06/2005 00:05:03 Successfully removed drbd resource
/etc/cluster/.resources/drbd/1.samba.sambavg.shares (file) from node sl4s1
INFO 13/06/2005 00:05:04 Successfully removed drbd resource
/etc/cluster/.resources/drbd/1.samba.sambavg.shares (file) from node sl4s2
INFO 13/06/2005 00:05:04 Successfully removed drbd resource
/etc/cluster/.resources/drbd/11.samba.sambavg.logs (file) from node sl4s2
INFO 13/06/2005 00:05:04 Successfully removed drbd resource
/etc/cluster/.resources/drbd/10.samba.sambavg.cfg (file) from node sl4s2
INFO 13/06/2005 00:05:04 Released port resource 9902 on node sl4s1
INFO 13/06/2005 00:05:04 Released port resource 9903 on node sl4s1
INFO 13/06/2005 00:05:04 Released port resource 9904 on node sl4s1
INFO 13/06/2005 00:05:04 Released port resource 9902 on node sl4s2
INFO 13/06/2005 00:05:05 Released port resource 9903 on node sl4s2
INFO 13/06/2005 00:05:05 Released port resource 9904 on node sl4s2
INFO 13/06/2005 00:05:05 Removed the fsmap entry from node sl4s1
INFO 13/06/2005 00:05:06 Removed the fsmap entry from node sl4s2
INFO 13/06/2005 00:05:06 Removing build checksums for samba
INFO 13/06/2005 00:05:06 Successfully removed application samba from the cluster.

```

Following this command the administrator is recommended to use the "clstat" to check to see if the application has indeed been removed from the cluster:

```

# clstat
Cluster: sl4cluster - UP

      Node      Status
      sl4s1     UP
      sl4s2     UP

Application   Node      State  Started  Monitor  Stale  Fail-over?
      apache   sl4s1   STARTED 0:00:23  Running    0      Yes

```

13 Stopping the Cluster

Although stopping the cluster is very straightforward the administrator is encouraged to read this section carefully. It contains information pertaining to the steps necessary to halt the cluster “manually”.

13.1 Stopping the Cluster Manually

The steps necessary to perform a manual cluster shut-down are very straightforward and consist of the following steps:

1. Stop all running applications
The first step is to stop all the running applications by issuing the relevant “clhaltapp” commands, an example of which was shown on page 84.
2. Kill of the cluster daemons on each host
Once all running applications have been stopped the next step is to kill of the cluster daemons running on each node. Firstly you need to know the name of your cluster – if you do not then use the “clstat” command:

```
# clstat
```

The output generated should be something like:

```
Cluster: sl4cluster - UP
```

```
Node      Status
sl4s1     UP
sl4s2     UP
```

```
Application  Node      State  Started  Monitor  Stale  Fail-over?
apache       N/A      DOWN   N/A      N/A      N/A    Yes
```

Once the name of the cluster is known - “sl4cluster” in this case, you need to look for a processes with the suffix “-*clustername*” and remove it from both machines:

```
# ps -ef | grep -- -sl4cluster
root      6190      1  0 Jun12 ?        00:00:28 cldaemon-sl4cluster
root      6193      1  0 Jun12 ?        00:00:00 cllockd-sl4cluster
root      6197      1  0 Jun12 ?        00:00:34 clnetd-sl4cluster
root      7511     7140  0 00:12 pts/1    00:00:00 grep -- -sl4cluster
```

Kill off this processes with the normal (default) signal (ignoring the “grep” command of course):

```
# kill 6190 6193 6197
```

This process should now be repeated on the other node. Once completed on both nodes use the “clstat” utility again to ensure that the cluster is down:

```
# clstat
ERROR 15/06/2005 02:17:42 Cluster sl4cluster is not running.
```

13.2 Stopping the Cluster Automatically

The utility called “clhalt” is provided for an even easier way to stop the cluster. If no applications are running all cluster daemons can be stopped using the command:

```
# clhalt
```

This will communicate with both nodes (if connectivity between both exist), and stop all daemons. This command will abort with an error if any applications are currently running. If no applications are running the output shown will be:

```
INFO 15/06/2005 02:18:50 Validated checksum for cluster configuration.
INFO 15/06/2005 02:18:50 Attempting to halt cluster sl4cluster...
INFO 15/06/2005 02:18:50 Attempting to contact a cluster daemon...
INFO 15/06/2005 02:18:50 Connecting to cluster daemon via host sl4s1
INFO 15/06/2005 02:18:50 Asking cluster daemons to abort...
INFO 15/06/2005 02:18:53 Cluster daemons aborted - cluster sl4cluster is DOWN.
```

If any applicatinos are running the default action is to abort the process of shutting down the cluster.

```
# clhalt
INFO 15/06/2005 02:21:02 Validated checksum for cluster configuration.
INFO 15/06/2005 02:21:02 Attempting to halt cluster sl4cluster...
INFO 15/06/2005 02:21:02 Attempting to contact a cluster daemon...
INFO 15/06/2005 02:21:02 Connecting to cluster daemon via host sl4s1
ERROR 15/06/2005 02:21:02 Unable to stop cluster - 1 application is still running
```

So unless the applications have been stopped the cluster will not halt. However the utility also supports a "--force" option which (by default) will stop any running applications and then shut the cluster down, as the following example shows:

```
# clhalt --force
INFO 15/06/2005 02:21:36 Validated checksum for cluster configuration.
INFO 15/06/2005 02:21:36 Attempting to halt cluster sl4cluster...
INFO 15/06/2005 02:21:36 Attempting to contact a cluster daemon...
INFO 15/06/2005 02:21:36 Connecting to cluster daemon via host sl4s1
INFO 15/06/2005 02:21:36 Attempting to stop application "apache" on sl4s1...
INFO 15/06/2005 02:21:41 Application "apache" has been halted successfully.
INFO 15/06/2005 02:21:41 Asking cluster daemons to abort...
INFO 15/06/2005 02:21:44 Cluster daemons aborted - cluster sl4cluster is DOWN.
```

The command always generates output to standard output - no "--verbose" option is required or supported. The information regarding the shut-down of the application in detail can be found in the application shut-down logs, available in the following directory and name format:

```
/var/log/cluster/clhalt.application.log
```

13.3 Halting Individual Nodes

The clhalt command also provides one other facility - the ability to stop the cluster services running on a specified node only. To do this the "--node" option is provided.

The administrator should note that the DRBD devices on the node removed from the cluster **are not stopped**. This means that data synchronisation will continue if possible.

By stopping the cluster daemons on a specified node it will ensure no applications can fail-over to this node until it rejoins the cluster. It also means that any application with stale data on the node taken out of the cluster will not start new attempts to synchronise that data.

Hence to stop node "serverb" from being part of the cluster:

```
# clhalt --node serverb
```

If that node is not running any applications the following output will be shown:

```
INFO 15/06/2005 02:32:10 Validated checksum for cluster configuration.
INFO 15/06/2005 02:32:10 Attempting to halt cluster sl4cluster...
INFO 15/06/2005 02:32:10 Aborting cluster daemon on sl4s2...
INFO 15/06/2005 02:32:11 Cluster Daemon halted on sl4s2.
```

If applications are running on the node then the default behaviour is to abort the node removal:

```
INFO 15/06/2005 02:34:09 Validated checksum for cluster configuration.
INFO 15/06/2005 02:34:09 Attempting to halt cluster sl4cluster...
ERROR 15/06/2005 02:34:09 Unable to remove node sl4s2 from cluster - it has
ERROR 15/06/2005 02:34:09 1 applications still running.
```

The behaviour can be changed by using the "--action" option, which can be set to one of the following:

- **error** (Default). If Any applications are currently running on the node in question then the removal of the node from the cluster will be aborted.
- **halt** All applications running on this node are halt prior to removing the node from the cluster.
- **failover** All applications running on this node are failed-over to the remaining node (if possible), before removing this node from the cluster.

For example to halt node "server1" ensuring any applications on it are failed-over use the following command:

```
# clhalt --node server1 --action=failover
INFO 25/06/2005 11:02:28 Validated checksum for cluster configuration.
INFO 25/06/2005 11:02:28 Attempting to halt cluster sl4cluster...
INFO 25/06/2005 11:02:28 Halting application apache on server1...
INFO 25/06/2005 11:02:34 Application apache halted in 6 seconds.
INFO 25/06/2005 11:02:36 SSH communication to server2 will be:
INFO 25/06/2005 11:02:36 192.168.100.34 ("drbd" network)
INFO 25/06/2005 11:02:36 Application apache starting on node server2...
INFO 25/06/2005 11:02:45 Application apache started in 9 seconds.
INFO 25/06/2005 11:02:45 Aborting cluster daemon on server1...
INFO 25/06/2005 11:02:46 Cluster Daemon halted on server1.
```

Each application will be failed-over separately, and once complete the node in question will be removed from the cluster.

13.4 Adding Nodes to a Running Cluster

If an individual node has been previously removed (or failed due to a hardware problem), once it has been rebooted or other actions taken so that it is ready to be added back to the cluster, then simply run the following command on either node:

```
# clform --join
INFO 25/06/2005 11:46:13 Validated checksum for cluster configuration
INFO 25/06/2005 11:46:13 SSH communication to server2 will be:
INFO 25/06/2005 11:46:13 192.168.100.34 ("drbd" network)
INFO 25/06/2005 11:46:13 Checking cluster status...
INFO 25/06/2005 11:46:13 sl4s2 is running - server1 will attempt to join cluster.
INFO 25/06/2005 11:46:13 Forced Join of cldaemon on server1...
INFO 25/06/2005 11:46:13 Waiting for server1 to join the cluster...
INFO 25/06/2005 11:46:20 Node server1 successfully joined cluster.
```

See the details on page 141 for further information on available options for this command.

14 Adding further Applications

14.1 Purpose of this section

The aim of this section is to introduce another application into the cluster whilst the cluster is already running. The ability to perform such changes was early in the development of Linuxha.net and thus is considered very robust. This is just one of the features the software offers in an attempt to minimise any requirement for cluster to application down-time for maintenance.

Since many of the steps here are almost identical to those presented for the initial sample application installation the section is quite brief. Only differences from the previous installation are explored in detail.

The application that is being added to the cluster is "samba" - this will obviously be running a Samba server hosting a couple of "shares". This is a very good example of the Linuxha.net software being used to provide high availability for storage for Windows® clients.

The specified sample files used in this section are available as a package from the following URL:

<http://www.linuxha.net/index.pl?ARGS=findproject:linuxha-samba,6>

Once the application has been downloaded to a temporary directory, the "tarp" package can be installed using the following command:

```
# tpinstall -i -p linuxha_samba -v
```

Administrators can alternatively install the RPM or Autopackage version of the sample application as required.

Once this has been installed the majority of the files necessary to build the sample application can be found in under the "/etc/cluster/samba" directory. This package only needs to be installed on the primary machine in the cluster. In the directory "/etc/cluster/samba/build" you will find a script to configure the environment. It is recommended that you utilise this script, or examine it and perform the necessary actions manually in your environment.

During the configuration of the environment all the steps below should be run as "root" on "serverA" only unless explicitly stated otherwise.

14.2 Application Storage Requirements

In this example we will create three file systems for use in a similar manner as for the "apache" package. The names, purposes and size of each logical volume is now explained:

Logical volume (Mount)	Size	Purpose
cfg (/samba/cfg)	20 Mb	Samba configuration and cluster application administration scripts
logs (/samba/logs)	20 Mb	Where the log files will reside for this copy of the smbd / nmbd processes.
shares (/samba/shares)	152 Mb	Where the shares will reside.

Of course the sizes of the "shares" area is far too small for a practical application - in reality the administrator would be expected to size this volume 10 or more Gigabytes. The size of 152Mb was chosen since the physical extent size for the volume group has been defined as 4Mb and this is large enough to serve sample files to a range of sample applications.

Please note that the sizes of the logical volumes might have been even smaller - but unfortunately most Linux file systems only work on sizes larger than 16Mb, (though Reiserfs actually requires at least 50Mb – at least on the test environments in use).

The commands necessary to create the volume group and logical volumes for the sample “Samba” configuration are as follows, (though the administrator will probably require different disk names and volume sizes):

```
vgcreate -s 8 /dev/app02vg /dev/sda7
lvcreate -L 20 -n cfg /dev/app02vg
lvcreate -L 20 -n logs /dev/app02vg
lvcreate -L 152 -n shares /dev/app02vg
```

If the administrator has used the supplied build script then apart from the volume group creation on both nodes, the logical volumes will have been created automatically. The “-s 8” argument to “vgcreate” ensures that logical volume sizes of around 500Gb are supported.

When using the build script please be aware of the file system configured to build. Typically this is configured as “jfs” and certain distributions, (such as Redhat Enterprise Linux 4), do not include support for this. In such cases change the configuration file to use “ext3” or if possible “xfs” instead.

Once these commands have been run successfully (on “servera”), the next step is to create the necessary file systems (if using jfs):

```
# mkfs -t jfs /dev/app02vg/cfg
# mkfs -t jfs /dev/app02vg/logs
# mkfs -t jfs /dev/app02vg/shares
```

Now the mount points should be created and the file systems mounted:

```
# mkdir -p /samba/cfg /samba/logs /samba/shares
# mount -t jfs /dev/app02vg/cfg /samba/cfg
# mount -t jfs /dev/app02vg/logs /samba/logs
# mount -t jfs /dev/app02vg/shares /samba/shares
```

Some sub-directories that will be used for the Samba service must now be created:

```
# mkdir /samba/logs/locks
# mkdir /samba/cfg/private
# mkdir /samba/shares/tmp
# mkdir /samba/shares/joe
```

14.3 Application Configuration for cluster

Now that the empty file systems are available typically the administrator would copy across the relevant production data to “/samba/shares” and handle configuration of “smb.conf” - the file that controls the Samba configuration.

For testing a special “smb.conf” is provided as part of the “linuxha_samba” package. This is a very cut down configuration just suitable for the example application - a realistic one will undoubtedly be much more complex.

Obviously the portion of the file concerned with networking is particularly important. The administrator must define the subnet to listen on and also ensure that the listening occurs on a particular IP address **only** - otherwise IP and node fail-over are unlikely to work.

The configuration file should be saved as the following file:

```
/samba/cfg/smb.conf
```

The file will be broken down and explained (though for a proper explanation please review the manual pages and documentation that the Samba project makes available).

The first part of the configuration file is where the global settings go - these are settings that control the Smb daemon overall, rather than relating to any particular share that it makes available. In this simple example the configuration will look as follows:

```
[global]
socket address = 172.16.177.50
interfaces = 172.16.177.0/255.255.255.0
bind interfaces only = yes
workgroup = LINUXHA
netbios name = linuxhapc
server string = linuxha server
security = user
encrypt passwords = yes
load printers = no
socket options = TCP_NODELAY
# Define file locations we intend to use
log file = /samba/logs/log.smb
log level = 1
max log size = 5000
lock directory = /samba/logs/locks
smb passwd file = /samba/cfg/private/smbpasswd
local master = yes
preferred master = yes
dns proxy = no
```

The first point to keep in mind is the “socket address” setting - this is set to the IP address of the application we intend to put in the cluster, and this should not be an address that already exists for any other application in the cluster, or statically on either of the servers in the cluster.

Much of the rest of the globals section is fairly standard (though the “workgroup”, “netbios” and “server string” names should be set to suitable values to allow us to recognise this Samba instance when browsing).

Please notice that near the end of the file the administrator must define locations for files that Samba uses. It is important that all these files are defined under directories that are part of the replicated data file systems - hence the reason that the “log file”, “lock directory” and “smb password file” settings all refer to directories under the “/samba” directory tree, using one of the appropriate file systems defined in the cluster.

In this case since it is only a example environment this section just defines two dummy shares, one for a shared temporary area, the other serving Joe's home directory.

```
[tmp]
comment = Temporary file space
path = /samba/shares/tmp
read only = no
public = yes
```

```
[jhome]
comment = Joes home dir Service
path = /samba/shares/joe
valid users = joe
public = no
writable = yes
```

14.4 Start and Stop scripts for “Samba” Application

In this instance the scripts to start and stop the “smbd” daemon process required will reside in the following directory:

```
/samba/cfg/scripts
```

The scripts, called “start” and “stop” will have the following content respectively:

```
#!/bin/sh

/usr/sbin/smbd -s /samba/cfg/smb.conf -D
exit $?
```

and:

```
#!/bin/sh

pid=`ps -eo pid,cmd | awk '$2 == "/usr/sbin/smbd" {print $1}`

if [ -n "$pid" ]
then
    kill $pid
    exit 0
else
    echo "Error: Unable to find process to kill" >&2
    exit 1
fi
```

It is also recommended that a “restart” script is also created which will be used by the process monitor to stop and start “smbd” if this daemon appears to fail. The contents of this script are:

```
#!/bin/sh

/samba/cfg/scripts/stop
sleep 1
/samba/cfg/scripts/start
exit $?
```

14.5 User Environment

Since the “tmp” share is globally available the administrator is recommended to set the permissions to something suitable:

```
# chmod 1777 /samba/shares/tmp
```

The above command also sets the “sticky” bit on the directory ensuring only the owner of files is able to delete them, (though anyone can actually truncate them and overwrite them!).

Joe’s home directory also needs correct permissions setting. Firstly the administrator must ensure that the “Joe” exists on both “ServerA” and “ServerB”, but running the following command (assuming this is a test environment).

```
# useradd -u 1001 joe
```

It does not matter what UID is used, **but it must be the same on both servers** - if it is not the mapping of data between one server and the other will not work when a fail-over occurs.

The administrator must also ensure that the following command is run on “ServerA” to create a Samba user for “joe”:

```
# smbpasswd -c /samba/cfg/smb.conf -a -U joe
```

Following this the administrator should set the local UNIX password for the “joe” account to the same values - on both servers ideally.

```
# passwd joe
```

14.6 Cluster Configuration

If the “build” script from the “linuxha_samba” package has not been used the administrator must create the directory for Samba configuration information on “ServerA”:

```
# mkdir /etc/cluster/samba
```

In this directory the “appconf.xml” must exist. For a sample configuration the contents may appear similar to the following:

```

<?xml version="1.0"?>
<appconf>
  <global>
    <name>samba</name>
    <version>0.2</version>
    <takeover>normal</takeover>
    <syncrate>5000</syncrate>
    <preferred_node>LEAST_CPU_LOAD</preferred_node>
    <!-- <dependencies></dependencies> -->
    <autostart>no</autostart>
  </global>

  <networks>
    <network net="prod" ip="192.168.0.86"
      broadcast="255.255.255.255"
      checklist="192.168.0.1,192.168.0.4" checkpercent="50"
      pingtype="icmp" pingtimeout="2"/>
  </networks>

  <vg>
    <name>sambavg</name>
    <type>filesystems</type>
  </vg>

  <application>
    <startscript>/samba/cfg/scripts/start</startscript>
    <stopscript>/samba/cfg/scripts/stop</stopscript>
    <maxstoptime>10</maxstoptime>
    <maxstarttime>10</maxstarttime>
  </application>
</appconf>

```

If the build script has been used from the sample package the only details likely to change are the networking information (probably just the IP address), and the volume group name.

Also in the “/etc/cluster/samba” directory on “servera” a Lems configuration file should be created. The name of this file should be:

```
/etc/cluster/samba/lems.local.xml
```

In theory it is possible to use a different name than “lems.local.xml”, but that option has not been validated for version 1.0.0.

```

<?xml version="1.0"?>
<lems_config>
  <globals modules="/sbin/cluster/lems/modules"
    programs="/sbin/cluster/lems/programs"
    logs="/var/log/cluster/lems"
    port="8801"
  />
  <check>
    <name>flag_check</name>
    <type>internal</type>
    <module>flag_check samba</module>
    <interval>10</interval>
    <action_list>
      <action rc="0" action="NOP"/>
      <action rc="1" action="%RCDATA%"/>
      <action rc="2" action="ABORT"/>
    </action_list>
  </check>
  <check>
    <name>smbd</name>
    <type>internal</type>
    <module>procmon /etc/cluster/samba/smbd.xml</module>
    <interval>10</interval>
    <action_list>
      <action rc="0" action="NOP"/>
      <action rc="1" action="STOP"/>
      <action rc="2" action="FAILOVER"/>
    </action_list>
  </check>

```

```

</check>
<check>
  <name>ip</name>
  <type>internal</type>
  <module>ip_module samba</module>
  <interval>10</interval>
  <action_list>
    <action rc="0" action="NOP"/>
    <action rc="1" action="RUN move_ip"/>
    <action rc="2" action="STOP"/>
  </action_list>
</check>
<check>
  <name>fsmonitor</name>
  <type>internal</type>
  <module>fsmon samba</module>
  <interval>10</interval>
  <action_list>
    <action rc="0" action="NOP"/>
    <action rc="1" action="PAUSE 30"/>
    <action rc="10" action="PAUSE 60"/>
    <action rc="2" action="STOP"/>
    <action rc="3" action="FAILOVER"/>
  </action_list>
</check>
</lems_config>

```

14.7 Checking the new Application Configuration

The steps to follow when adding additional applications to an existing cluster are exactly the same as when adding the first application, and can be done when the cluster is running or shut-down.

The steps must be able to communicate to both nodes and so applications can only be defined or changed when both nodes are running and networked in a production-like manner.

Thus to check the application configuration the administrator should run the following on "servera":

```
# clbuildapp --application samba --verbose --check
```

When this is run the output it generates should appear similar to the following:

```

INFO 09/06/2005 01:27:19 Backups directory defaulted to /clbackup
INFO 09/06/2005 01:27:20 Global section configuration validation complete
INFO 09/06/2005 01:27:20 Checked that node names resolve to IP addresses
INFO 09/06/2005 01:27:20 Successfully ssh'd from sl4s1 to sl4s2
INFO 09/06/2005 01:27:21 Successfully ssh'd from sl4s2 to sl4s1
INFO 09/06/2005 01:27:22 Attempting to backup cluster configuration on node sl4s1.
INFO 09/06/2005 01:27:22 Saved cluster configuration in 37879 bytes on node sl4s1
INFO 09/06/2005 01:27:22 Attempting to backup cluster configuration on node sl4s2.
INFO 09/06/2005 01:27:23 Saved cluster configuration in 9488 bytes on node sl4s2
INFO 09/06/2005 01:27:23 md5sum for sl4s1 is: /usr/bin/md5sum
INFO 09/06/2005 01:27:23 md5sum for sl4s2 is: /usr/bin/md5sum
INFO 09/06/2005 01:27:23 Loading DRBD kernel module locally.
INFO 09/06/2005 01:27:25 Loading DRBD kernel module on remote host.
INFO 09/06/2005 01:27:27 Validated clconf.xml is consistent on both nodes
INFO 09/06/2005 01:27:27 Validated mkdir cluster utility on sl4s1
INFO 09/06/2005 01:27:28 Validated mkdir cluster utility on sl4s2
INFO 09/06/2005 01:27:28 Validated mkmdcfg cluster utility on sl4s1
INFO 09/06/2005 01:27:28 Validated mkmdcfg cluster utility on sl4s2
INFO 09/06/2005 01:27:28 Validated drbd_tool cluster utility on sl4s1
INFO 09/06/2005 01:27:28 Validated drbd_tool cluster utility on sl4s2
INFO 09/06/2005 01:27:28 Validated specified syncrate setting is sane.
INFO 09/06/2005 01:27:28 Autostart parameter present and valid.
INFO 09/06/2005 01:27:28 Preferred_node parameter present and valid.
INFO 09/06/2005 01:27:28 Current status of application samba is: NOT_REGISTERED
INFO 09/06/2005 01:27:28 Number of networks to validate: 1
INFO 09/06/2005 01:27:28 Validated single IP address for application
INFO 09/06/2005 01:27:28 Validated network prod defined in cluster topology.
WARN 09/06/2005 01:27:28 Network prod missing netmask - will use default.

```

```
INFO 09/06/2005 01:27:28 Validated optional broadcast for prod is in required format.  
INFO 09/06/2005 01:27:28 Validated network prod attributes.  
INFO 09/06/2005 01:27:28 Status directory on sl4s1 already exists.  
INFO 09/06/2005 01:27:29 Validated/created status directory on sl4s2
```

Once the application configuration has been checked then the “Lems” session information should be next. This can be checked by running the following command:

```
# lems.pl --application samba \
  --config /etc/cluster/samba/lems.local.xml \
  --verbose --file /dev/tty --check
```

This should then output something similar to the current terminal:

```
INFO 09/06/2005 01:28:16 Using modules from : /sbin/cluster/lems/modules
INFO 09/06/2005 01:28:16 Using programs from : /sbin/cluster/lems/programs
INFO 09/06/2005 01:28:16 Writing logs to : /var/log/cluster/lems
INFO 09/06/2005 01:28:16 Validating port not open locally...
INFO 09/06/2005 01:28:16 Validating port not open on sl4s2...
INFO 09/06/2005 01:28:16 No clash for application apache - uses port 8800 not 8801.
INFO 09/06/2005 01:28:16 Listening on port : 8801
INFO 09/06/2005 01:28:16 Global initialisation complete.
INFO 09/06/2005 01:28:16 Started local server on port 8801
INFO 09/06/2005 01:28:16 Validating monitor entry ip...
INFO 09/06/2005 01:28:16 Validated monitor entry ip successfully.
INFO 09/06/2005 01:28:16 Validating monitor entry fsmonitor...
INFO 09/06/2005 01:28:16 Validated monitor entry fsmonitor successfully.
INFO 09/06/2005 01:28:16 Validating monitor entry smbd...
INFO 09/06/2005 01:28:16 Validated monitor entry smbd successfully.
INFO 09/06/2005 01:28:16 Validating monitor entry flag_check...
INFO 09/06/2005 01:28:16 Validated monitor entry flag_check successfully.
INFO 09/06/2005 01:28:16 Check mode - transferring validated config to remote node.
WARN 09/06/2005 01:28:17 Unable to transfer configuration to remote node.
INFO 09/06/2005 01:28:17 Calculated a check interval of 5.0 seconds.
```

Notice in the output the third line from the bottom - when running in “--check” mode the configuration file that has been successfully checked will be automatically copied to the other node in the cluster. This has one important impact on possible “Lems” configurations:

If the administrator wishes to use a different configuration file on each host for Lems that configuration should be copied back to the correct location following the successful build of the application..

14.8 Allocating Application Resources

As before following the validation of the application configuration file, and the Lems configuration, the next step is to allocate the required resources to the application.

If the application was built using the “buildit” script from the sample “linuxha_samba” package, then firstly un-mount the file systems on the 2nd node:

```
# umount /samba/cfg
# umount /samba/logs
# umount /samba/shares
```

The file systems must remain mounted on the primary node from where the “clbuildapp” commands are run. Volume group configuration is done using the “--vgbuild” option:

```
# clbuildapp --application samba --vgbuild --verbose
```

This will ensure that the volume group exists on both sides of the cluster - aborting if it does not. Any logical volumes defined locally in the specified volume group will be checked and created on the remote. Both hosts will also have additional logical volumes added to account for the meta-data for DRBD device management. The output below has been shorted to remove duplicate output that was seen during the “--check” option.

Of course if you have made use of the “build” script from the “linuxha-samba” package, the logical volumes will already have been made. Even so you must perform this step to ensure a valid configuration – otherwise you will not be able to allocate the resources for the application in the cluster.

```

INFO 09/06/2005 01:32:38 Status directory on sl4s1 already exists.
INFO 09/06/2005 01:32:38 Validated/created status directory on sl4s2
INFO 09/06/2005 01:32:38 Getting list of defined volume groups on sl4s2
INFO 09/06/2005 01:32:39 Validated Meta volume exists for shares (and is 128mb)
INFO 09/06/2005 01:32:39 Validated Meta volume exists for cfg (and is 128mb)
INFO 09/06/2005 01:32:39 Validated Meta volume exists for logs (and is 128mb)
INFO 09/06/2005 01:32:40 VG sambavg does not contain LV cfg_meta on sl4s2 -
creating...
INFO 09/06/2005 01:32:41 VG/LV sambavg/cfg_meta of 131072Kb built on sl4s2
INFO 09/06/2005 01:32:41 VG sambavg does not contain LV logs_meta on sl4s2 -
creating...
INFO 09/06/2005 01:32:42 VG/LV sambavg/logs_meta of 131072Kb built on sl4s2
INFO 09/06/2005 01:32:42 VG sambavg does not contain LV shares_meta on sl4s2 -
creating...
INFO 09/06/2005 01:32:43 VG/LV sambavg/shares_meta of 131072Kb built on sl4s2
INFO 09/06/2005 01:32:43 All LV checks on VG sambavg completed
INFO 09/06/2005 01:32:44
INFO 09/06/2005 01:32:44 Now run with the --build to ensure all required resources
INFO 09/06/2005 01:32:44 are allocated on both nodes
INFO 09/06/2005 01:32:44

```

Following the successful Volume Group configuration the actual cluster server resources, (devices and ports), should now be allocated. To do this the administrator should run the following command on "servera":

```
# clbuildapp --application samba --build --verbose
```

The interesting lines of output this will generate are shown below - notice the allocation of resources to the application.

```

INFO 09/06/2005 01:33:39 Validated VG build run has completed against this
configuration.
INFO 09/06/2005 01:33:39 LVM on sl4s1 appears to be version 2
INFO 09/06/2005 01:33:39 LVM on sl4s2 appears to be version 2
INFO 09/06/2005 01:33:39 Checked volume groups exist on sl4s1
INFO 09/06/2005 01:33:40 Checked volume groups exist on sl4s2
INFO 09/06/2005 01:33:40 Checked 6 logical volumes for sambavg on sl4s1
INFO 09/06/2005 01:33:40 Completed volume group analysis on sl4s1:
INFO 09/06/2005 01:33:40 VG: sambavg Used LVs: 3 Not Used LVs: 0
INFO 09/06/2005 01:33:40 sl4s1 available: PORTS: 98, DRBD: 49
INFO 09/06/2005 01:33:40 sl4s2 available: PORTS: 98, DRBD: 49
INFO 09/06/2005 01:33:40 Validated enough resources available on both nodes
INFO 09/06/2005 01:33:40 Host/VG/LV sl4s1/sambavg/shares allocated port 9902
INFO 09/06/2005 01:33:41 Host/VG/LV sl4s2/sambavg/shares allocated port 9902
INFO 09/06/2005 01:33:41 Host/VG/LV sl4s1/sambavg/shares allocated DRBD 1
INFO 09/06/2005 01:33:42 Host/VG/LV sl4s2/sambavg/shares allocated DRBD 1
INFO 09/06/2005 01:33:42 Host/VG/LV sl4s1/sambavg/cfg allocated port 9903
INFO 09/06/2005 01:33:42 Host/VG/LV sl4s2/sambavg/cfg allocated port 9903
INFO 09/06/2005 01:33:42 Host/VG/LV sl4s1/sambavg/cfg allocated DRBD 10
INFO 09/06/2005 01:33:43 Host/VG/LV sl4s2/sambavg/cfg allocated DRBD 10
INFO 09/06/2005 01:33:43 Host/VG/LV sl4s1/sambavg/logs allocated port 9904
INFO 09/06/2005 01:33:44 Host/VG/LV sl4s2/sambavg/logs allocated port 9904
INFO 09/06/2005 01:33:44 Host/VG/LV sl4s1/sambavg/logs allocated DRBD 11
INFO 09/06/2005 01:33:45 Host/VG/LV sl4s2/sambavg/logs allocated DRBD 11
INFO 09/06/2005 01:33:45 Created directory /etc/cluster/.status/samba on node sl4s1
INFO 09/06/2005 01:33:45 Validated/created application status directory on sl4s2
INFO 09/06/2005 01:33:45 Validated/created fsmap directory on sl4s1
INFO 09/06/2005 01:33:45 Validated/created fsmap directory on sl4s2
INFO 09/06/2005 01:33:45 Validated/created application config directory on sl4s2
INFO 09/06/2005 01:33:46 Copied config data for samba to sl4s2
WARN 09/06/2005 01:33:47 Following problems for path /samba/cfg consistency:
WARN 09/06/2005 01:33:47 Permissions differ for //samba/cfg [local=0777,remote=0755]
INFO 09/06/2005 01:33:48 Validated file system mount point /samba/logs
INFO 09/06/2005 01:33:49 Validated file system mount point /samba/shares
INFO 09/06/2005 01:33:49 Created fsmap data for samba on sl4s1
INFO 09/06/2005 01:33:49 Copied fsmap data for samba to sl4s2
INFO 09/06/2005 01:33:50 Deleted TIME file from node sl4s2
INFO 09/06/2005 01:33:50 Application samba now (re)registered with cluster.
INFO 09/06/2005 01:33:50 Transferring XML file lems.local.xml to sl4s2.
INFO 09/06/2005 01:33:50 Transferring XML file smbd.xml to sl4s2.
INFO 09/06/2005 01:33:51
INFO 09/06/2005 01:33:51 Cluster build has completed successfully.
INFO 09/06/2005 01:33:51 Now use the --sync option to ensure that all

```



```
INFO 09/06/2005 01:33:51 file systems on the other node are synchronised
INFO 09/06/2005 01:33:51
```

Please notice that the output included the following warning:

```
WARN 09/06/2005 01:33:47 Following problems for path /samba/cfg consistency:
WARN 09/06/2005 01:33:47 Permissions differ for //samba/cfg [local=0777,remote=0755]
```

It is recommended that the cause of this warning is corrected before continuing – such warnings are given for good reason.

Remember that if any permissions (and user/group) on any component of any path that is used to mount a replicated directory is different on both servers it may result in the application running differently on each server.

In this case the “local” permissions should be changed to ensure they match those on “serverb”:

```
# chmod 777 /samba/shares
```

At this point all necessary resources have been allocated and as the output states the final part of adding the application to the cluster is to synchronise the data to ensure we have two valid copies of the data.

This is done using the following command:

```
# clbuildapp --application samba --sync --verbose
```

This will generate output culminating with the data synchronisation. Only when the synchronisation is complete will the script exit:

```
INFO 09/06/2005 01:35:27 Loading details of file systems to unmount
INFO 09/06/2005 01:35:27 File system /samba/cfg un-mounted
INFO 09/06/2005 01:35:27 File system /samba/logs un-mounted
INFO 09/06/2005 01:35:27 File system /samba/shares un-mounted
INFO 09/06/2005 01:35:27 All file systems un-mounted successfully.
INFO 09/06/2005 01:35:28 Some/all drbd devices need configuration on sl4s1
INFO 09/06/2005 01:35:29 All drbd services now running on sl4s1
INFO 09/06/2005 01:35:31 Attempted to start new drbd services on sl4s2
INFO 09/06/2005 01:35:31 All drbd services now running on sl4s2
INFO 09/06/2005 01:35:32 Successfully started rebuild of DRBD devices for samba.
INFO 09/06/2005 01:35:32 Sync status: Devices sync'ed: 0, Devices unsync'ed: 0
INFO 09/06/2005 01:35:32 Sync status: Currently syncing LV /dev/sambavg/shares, 0.0%
completed
INFO 09/06/2005 01:35:42 Sync status: Devices sync'ed: 2, Devices unsync'ed: 0
INFO 09/06/2005 01:35:42 Sync status: Currently syncing LV /dev/sambavg/shares, 39.5%
completed
INFO 09/06/2005 01:35:52 Sync status: Devices sync'ed: 2, Devices unsync'ed: 0
INFO 09/06/2005 01:35:52 Sync status: Currently syncing LV /dev/sambavg/shares, 93.7%
completed
INFO 09/06/2005 01:35:57 Sync status: Devices sync'ed: 3, Devices unsync'ed: 0
INFO 09/06/2005 01:35:57 All logical volumes on remote host sl4s2 synchronised
INFO 09/06/2005 01:35:57 Stopping DRBD on sl4s1 devices.
INFO 09/06/2005 01:35:59 Stopping DRBD on sl4s2 devices.
INFO 09/06/2005 01:36:00
INFO 09/06/2005 01:36:00 Successfully Synchronised data from sl4s1 to sl4s2
INFO 09/06/2005 01:36:00
```

At this point the application is fully valid. Since the cluster daemons are currently running messages similar to the following will appear at the end of the log for the cluster daemon (if running in verbose mode):

```
INFO 09/06/2005 01:33:50 Validated Build for samba is valid - will register with
cluster.
INFO 09/06/2005 01:33:50 Application samba has been registered with the cluster
daemons
```

At this point running the “clstat” command will indicate that the application is now known to the cluster:

```
Cluster: sl4cluster - UP
```

Node	Status
sl4s1	UP
sl4s2	UP

Application	Node	State	Started	Monitor	Stale	Fail-over?
apache	N/A	DOWN	N/A	N/A	N/A	Yes
samba	N/A	DOWN	N/A	N/A	N/A	Yes

From this point onwards the administrator can start the application as required - there is no requirement to stop/start or “refresh” are cluster processes. As usual the application can be started using just:

```
# clrunapp --application samba
```

If any warnings were issued during the build phases these should be rectified before attempting application fail-over testing. For example, if the “build” script was not used for this sample configuration the administrator may need to create the file system mount points manually on “serverB”:

```
# ssh serverb mkdir -p /samba/cfg /samba/logs /samba/shares
```

Failure to ensure mount points exist on the other node will result in the application failing to start on “serverB” since it is unable to mount the required file systems.

14.9 Limitations with Samba Sample Application

The above application works - you would mount the temporary directory on a Linux host using a command similar to the following:

```
# mount -t smbfs -o username=joe //172.16.177.50/tmp /tmpmnt/
```

At this point failure of the host running the “samba” application will result in the application migrating to the remaining host. For the duration of the fail-over event any attempt at access to the resource will hang or error. Once the fail-over is completed access to the data provided by the share should work as normal again.

Since the example Samba application is very basic note that the “joe” account had to be created and managed as accounts on separate machines.

A better approach would be to manage the accounts via a clustered LDAP server instead. This would remove the requirement of having to keep the UNIX / SMB password files synchronised.

Production quality Samba distributions with Linuxha.net are possible (and are indeed already in operation). System administrators with more experience with Samba can undoubtedly provide improvements for this sample package to remove limitations of this sample implementation.

14.10 Adding Applications: Common Problems

Of course following the instructions given above should result in a perfectly working cluster application, (although with some limitations as explained previously). Despite attempting to simplify the build and management of clustered applications, problems can still occur. Some of the more common problems found when using applications with Linuxha.net are now explained.

14.10.1 Mismatch Security Settings

Most applications have the concept of a “software owner” - this is a user-account that is used both to own files and is expected to run common processes or daemons for that application. In such scenarios it is very important to use that the user ID in question exists on both nodes in the cluster.

Further the administrator should ensure that the UID and GID numbers map on both servers, and even the profiles which define the environment are the same. Even small differences can result in subtle problems following a fail-over.

For example having a different umask on each node could result in problems and be hard to spot. The administrator must take care to ensure the components that are **not shared** are compatible.

In a similar manner the permissions on mount points for directories in the application should also be the same on both nodes to ensure obscure differences between running the applications on either node do not occur. When building the application warnings are issued if problems are found; such warnings should not be ignored. Of course these checks only take place when an application is built, it is the responsibility of the administrator to ensure any changes outside of the replicated data areas are duplicated on both hosts.

14.10.2 Missing Mount Points

A common problem is that mount points for directories on the secondary node are simply not created when the application environment is built. Missing mount points are shown as warnings during the build process for an application. Failure to ignore this will result in non-functioning fail-over for the application in question.

However missing mount points or incorrect permissions are still possible since changes following the application build might have resulted in the mount point directories having been removed, renamed or simply their owner, group or permissions changed to become more restrictive.

Part III:
Cluster System Administration

15 System Administrator Responsibilities

Depending on the environment the system administrator may be responsible for on going support of the cluster, or may hand-over to those involved in a "support" role. Whoever is involved with day-to-day support of the cluster should read the information in this section.

From the author's experience many problems with cluster availability are not down to hardware or software failures, but due to errors made by support staff.

Thus it is critically important that whoever manages the cluster carefully reads this section. Ideally if the implementation and support roles are separate those involved in support should be involved in implementation as much as possible to smooth the transition of the cluster into a production environment.

The purpose of this part of the document is to describe the typical activities that might occur when managing a live cluster – and how they should be handled. Further this section contains information that should be considered if the configuration of the cluster needs to change at all – which it often does over the lifetime of the environment.

Importantly the final section concludes with various failure scenarios that might occur and the steps the administrator may need to following to allow client access to continue, or in the worst cases, to be recovered.

The sections here mirror typical responsibilities of the cluster administrator, including;

- **Managing Application Monitoring with "Lems"**
Lems monitoring is critical to the management of any application running in the cluster. Although these daemons will run without intervention for as long as necessary it is possible to communicate with them to ensure the application being monitored works in the manner expected.

Typically communication is required if the administrator wishes to manually stop the application processes, but not shutdown the complete cluster application. This is useful when performing software upgrades, for example.

Additionally the administrator may wish to change, remove or add additional monitors for an application, and does not want to stop the application to take advantages of the changes.
- **Checking Cluster and Application State Information**
The "clstat" output should be used both when the cluster is running normally to validate everything is indeed running as expected.

Checks should also take place to ensure all cluster processes are running as expected - to ensure processes such as "cldaemon" or "clnetd" are running on cluster nodes.
- **Adding / Removing Applications**
Although the process of adding applications has previously been dealt with some additional information regarding limitations and impact to existing cluster infrastructure are considered.

Given that some clusters have many applications it is inevitable that eventually an existing application will need to be removed. This section discusses how applications can be removed from the running cluster, and considers the impact of this activity on cluster resources and application availability – including LVM interaction.
- **Changing Cluster Configuration**
Although Linuxha.net does not implement limits in many places it is sometimes necessary to add further resources - such as the maximum number of devices to use.

Further this section indicates which configuration parameters defined in the cluster

topology configuration can be changed whilst the cluster is running, and the impact of such changes.

➤ **Changing Application Configurations**

Some applications are more complex than others. The more complex an application the greater the likelihood that some problem will be found with the environment once the cluster is "live". Hence Linuxha.net allows many of the specifics of the application configuration to be changed dynamically - though some work remains in this field, (such as IP address manipulation).

It is very unusual for the resources required for an application not to change over time. Databases and file servers often grow rapidly. Support on dynamic reconfiguration of disk resources assigned to an application is seen as a key benefit of Linuxha.net.

➤ **General systems administration**

Tasks such as changing the verbosity of logs, restarting individual daemons, and dealing with software upgrades must be dealt with.

➤ **Understanding current Linuxha.net Limitations**

Although Linuxha.net has been designed to ensure as many changes to the environment as possible do not require the cluster to be shut down, there are some limits to this "dynamic" nature of the application. All current limitations are described here, along with the steps necessary to make such off-line changes.

➤ **Handling Failure Scenarios**

The administrator is responsible for ensuring that the cluster continues to run without failure, but if a failure does occur the administrator must be able to quickly determine the reason for the failure and then take appropriate actions.

Sometimes the cluster is unable to recover automatically and so the administrator may be called to manually intervene to bring application back on-line, and hopefully plan changes to the future to avoid such scenarios occurring again.

16 Managing Application Monitoring

Lems has been introduced as the mechanism by which the individual applications in the cluster monitor the status of resources owned by that application and take relevant steps when changes are noted. The “change” might be the failure of an application, loss of data synchronisation, or even noting the presence of a file or flag.

Each check within Lems is actually implemented by separate modules. Thus it is possible for the administrator to add custom modules to a configuration relatively easily. Information covering the technical details of Lems, including information on standard modules and how to write custom modules can be found on page 174.

Management of a Lems daemon for a running application is one of the more typical actions the administrator might perform. One of the most common mistakes affecting application availability is incorrectly assuming Lems might not notice a change. For example if the administrator wishes to quickly stop a process, perform some actions and then restart it, the Lems daemon may notice the failure and attempt to restart the application. If several attempts occur in a short period of time it could cause an application fail-over.

Linuxha.net includes a utility called “lemsctl” to provide an interface for sending messages to a running Lems daemon. The typical syntax in use is very straightforward:

```
# lemsctl --application appname --msg "message to send"
```

The daemon itself understands several messages (as described next), and it is also possible to pass messages down to individual monitors that are currently running. As with all other network communications within Linuxha.net the transport of information is encoded using the cluster key, thus protecting against others attempting to use it - in the same manner as the cluster daemons themselves.

Of course since the key is stored in a file that is only readable by “root”, no other user accounts in either node in the cluster will be able to use this utility.

16.1 Stopping Monitoring

If the administrator wishes to change an application, but is unsure of the impact the change will have it is recommended that **all Lems monitors for the application are stopped**. This will ensure that the cluster software does not attempt to “repair” a situation that you may wish to correct manually.

To stop all Lems monitors for a particular application the following command can be used:

```
PAUSE
```

Hence to send them message to the Lems daemon currently running for the “apache” application use the following command:

```
# lemsctl --application apache --msg PAUSE
```

It does not matter which node this command is run on – it will ensure the request is sent to the correct server / daemon – assuming the specified application is actually running and has a functioning Lems daemon..

When this command has been run the detailed section of the output for an application that has “lems” running will show all monitors as “Stopped”:

```
# clstat -A apache
Cluster: sl4cluster - UP

Application      Node      State  Runnig  Monitor  Stale  Fail-over?
  apache         sl4s2    STARTED  0:00:05  Running    0         Yes

File Systems
```


Mount Point	Valid	Type	State	% Complete	Completion
/apache	both	drbd	Sync		

Process Monitors

Name	Status	Restarts	Current	Reset at
httpd	Stopped	3	0	N/A

General Monitors

Type	Name	Status
Flag Check	flag_check	Stopped
FS Monitor	fsmonitor	Stopped

Following this no resources are being monitored by the Lems daemon for “apache”. Any other Lems daemons for other running packages will continue unaffected. Thus all “software” changes can be performed against the application, though physical network and node checking continues (by the network and cluster daemons respectively).

16.2 Resuming Monitoring

If use of the “PAUSE” has been made and any changes are complete – the administrator is recommended to ensure that all monitoring for the application is started again as soon as possible. This can be achieved through the use of the “RESUME” command:

```
# lemsctl --application apache --msg RESUME
OK
```

Now checking the monitors should show something similar to:

```
# clstat -A apache
Cluster: sl4cluster - UP

Application      Node      State  Runnig  Monitor  Stale  Fail-over?
  apache        sl4s2    STARTED  0:00:05  Running    0      Yes

File Systems

Mount Point      Valid  Type  State  % Complete  Completion
/apache          both  drbd   Sync

Process Monitors

      Name  Status  Restarts  Current  Reset at
      httpd  Running    3         0        N/A

General Monitors

      Type      Name      Status
Flag Check  flag_check  Running
FS Monitor  fsmonitor  Running
```

The important point to note here is that if a module was “Stopped” prior to the “PAUSE” command, then the “RESUME” will leave it stopped – which is exactly what is needed – consider otherwise what would happen if the “IP fail-over” monitor was started at this point, for example.

16.3 Pausing a Module

Although stopping monitoring of all resources for an application is very useful, after some experience with the software it is better just to stop certain monitors. For example the administrator should not need to stop monitoring the status of data replication if changes are just being made to the application data.

Stopping certain monitors can actually be achieved in two different ways;

- *Flag Monitor* - this monitor is provided since it allows the administrator to specify a directory which can be checked for certain files and then modules stopped whilst the files exist. This is provided since it can be configured so the directory does not need “root” access to stop monitoring - useful when application administrator is performed by different users, rather than the cluster administrator.
- *Lemsctl Utility* - the facility recommended for the cluster administrator since unlike the “flag monitor” method it can be run on either machine (rather than on the machine running the application), and does not require the administrator to remember which directory to create the file in.

When stopping each monitor recall that the name of each is unique, and hence this name can be used to ensure just that particular monitor is stopped. For example using the monitors shown in the example on the previous page, to stop the “httpd” process monitor the administrator could use the following command:

```
# lemsctl --application apache --msg "PAUSE httpd"
httpd PAUSED
```

Now the “Process Monitors” section of the output would look similar to:

Process Monitors

Name	Status	Restarts	Current	Reset at
httpd	Stopped	3	0	N/A

Thus the administrator gives the name of the monitor, which will be different from the type of the monitor. This allows the same monitor to be used multiple times in a Lems configuration, and each to be handled individually by “lemsctl”.

16.4 Resuming a Module

Once the required administration work is complete it is recommended that monitoring is started as soon as possible. Starting monitoring for a particular monitor is also very straightforward – for example to restart the “httpd” monitor that was paused previously use:

```
# lemsctl --application apache --msg "RESUME httpd"
httpd RESUMED (in 10 secs)
```

Notice in this instance that monitor will resume checking after the check interval defined for the application has been passed.

16.5 Removing a Monitor

The previous actions of pausing and resuming monitors tend to be used for transient maintenance on the environment in which the cluster operates. However it is also possible to perform more permanent changes to the monitor configuration without actually stopping a package – adding further to the availability features of this product.

Taking this into account it is possible to modify the monitors that a running “lems” session uses without actually having to restart the “lems” daemon process or the application in the cluster that it monitors. Again access to this functionality is via the “lemsctl” command. Prior to removing an

existing monitor note the name of the monitor from the output of the “clstat” command, for example:

```
# clstat --application apache
```

Consider the monitor section of the output:

General Monitors

Type	Name	Status
Flag Check	flag_check	Running
FS Monitor	fsmonitor	Running

If the administrator wishes to remove the “Flag Check” monitor named “flag_check” then the following command could be used:

```
# lemsctl --application apache --msg "REMOVE flag_check"
flag_check REMOVED
```

Now the output from the “clstat” command will not include any details of that monitor, and the log file, (if verbose logging is enabled), should contain a line similar to:

```
INFO 09/06/2005 02:15:22 Removed monitor "flag_check" by user request.
```

After this point the specified monitor will no longer be part of the running Lems configuration for the application in question.

It is important to realise that this action only modifies the **running** configuration - if the administrator wishes to remove the monitor permanently the XML configuration file for the Lems daemon in question must be modified - **on both hosts**.

Once the specified monitor has been removed it can be modified (or its configuration changed in the XML configuration file - and then reloaded (see below) - without actually impacting the availability of the application being monitored in any way beyond the monitor in question.

Although it is possible to remove any monitor in this manner it is recommended that the administrator only remove custom monitors unless they have significant experience using the product.

The above point should be taken seriously - removing the “FS Monitor” is great as an example but could seriously impact the availability of the environment if left from the running configuration for a period of time⁸.

16.6 Adding a new Monitor

To compliment the support for removal of modules from the running configuration the “lemsctl” command also supports the ability to actually add a new monitor.

It is not possible to add a monitor that already configured - instead use the “REMOVE” facility as described in the previous section first.

To install a new module the administrator must get the name of the module from the configuration file, (in this case the intention is to reload the “flag_check” module), and pass this to the Lems daemon using the “INSTALL” command, for example:

```
# lemsctl --application apache --msg "INSTALL flag_check"
OK
```

Examining the “lems” log file now should show some text similar to the following:

```
INFO 09/06/2005 02:17:01 *** INSTALL MODULE flag_check START ***
```

⁸ The “Fs monitor” is used to automatically recover from unsynchronised data and thus not running it could impair the quality of the data for the application in question.

```
INFO 09/06/2005 02:17:01 New configuration using modules from :
/sbin/cluster/lems/modules
INFO 09/06/2005 02:17:01 New configuration using programs from :
/sbin/cluster/lems/programs
INFO 09/06/2005 02:17:01 Validated new monitor entry flag_check successfully.
INFO 09/06/2005 02:17:01 Successfully loaded module flag_check.pm
INFO 09/06/2005 02:17:01 Scheduled flag_check for 1118279826 (object type=flag_check)
INFO 09/06/2005 02:17:01 Successfully added new monitor flag_check.
INFO 09/06/2005 02:17:01 *** INSTALL MODULE flag_check END ***
```

16.7 Monitor Specific Communication

The commands described previously in this section are generic to any monitor that is currently in use. However a more fine-grained control mechanism for management of individual monitors is also available.

This is made possible via specific messages that each module implements. This functionality is optional, and will be different for each module. Information on how to write a module to handle specific requests in this manner see the technical information starting on page 188.

All custom messages are sent to the monitor in question using the following syntax form of the "lemsctl" command:

```
# lemsctl --application appname --msg "CMD monitor command"
```

The "command" can be one or more words, the format of which is determined by the particular monitor itself.

The process monitor is probably one of the most common monitors that an administrator might spend time sending messages to. Consider the following process monitor status report from "clstat":

Process Monitors

Name	Status	Restarts	Current	Reset at
httpd	Running	3	1	09/06/2005-03:18

Although the process monitor supports the ability for the number of current restarts of the application to be reset automatically after a given time period, it is often useful to reset this earlier - for example if the failure reason has been identified and fixed, and the administrator wishes to ensure that the fail-over scenario now reflects a "fresh" cluster configuration.

For this reason the "procmon" type monitor (Process Monitor) support a custom "RESET" option, which can specified using the custom message format of command:

```
# lemsctl -application apache -msg "CMD httpd RESET"
OK
```

The response offered gives an indication of whether the command was understood, and/or whether it was successful. The convention is that "OK" means that the command was understood and actioned accordingly.

Now when the process monitor information is examined the output should show:

Process Monitors

Name	Status	Restarts	Current	Reset at
httpd	Running	3	0	N/A

Another custom message accepted by "procmon" type monitors is "MAXCOUNT" - this can be used to change the maximum number of times an application can be restarted before a fail-over attempt is tried.

For example to set the "Restarts" value to 10, you could use the following command:

```
# lemsctl --application apache --msg "CMD httpd MAXCOUNT 10"
OK
```

After this the restart count for the monitor will be changed to 10 – simple!

Again as with previous "lemsctl" commands remember that any changes are temporary - the configuration files are not updated so if the changes are to be permanent then the administrator should modify the relevant XML file as well - on both nodes.

16.8 Log Management

Although all Linuxha.net daemons can operate with very little logging (just reporting on errors or warnings), it is recommended that the “verbose” logging options be used in all circumstances.

Log files are one of the main tools of any system administrator, but they can also become a real problem if they can not be managed easily. To ensure log files remain manageable the Lems daemon supports two directives related to logging - “VERBOSE” and “LOGCYCLE”. The first of these provides the ability to turn the logging verbose logging on or off:

```
# lemsctl --application apache --msg "VERBOSE OFF"
OK
```

This obviously turns verbose logging off – to turn it back on again, simply replace “OFF” with “ON”.

The second directive is more useful – it allows the log files to be cycled automatically. This is done using the “LOGCYCLE” directive. Consider having the following log file for the “apache” lems daemon:

```
-rw-r--r--  1 root  root          3825 Mar  9 09:34 lems-apache.log
```

Use of the “fuser” command will show that this file is open for writing, so changing the file directly via shell utilities, (such as renaming it, removing it or compressing it) is not recommended. However the “LOGCYCLE” directive is designed for this scenario, consider:

```
# lemsctl --application apache --msg "LOGCYCLE 5"
OK
```

Now the log file directory will instead contain:

```
-rw-r--r--  1 root  root          3825 Mar  9 09:34 lems-apache.log.000
-rw-r--r--  1 root  root           59 Mar  9 09:36 lems-apache.log
```

The older log file has been cycled to the “000” file. Running the same command again would cycle the logs up to the number specified in the command. Currently it is possible to specify a number between 1 and 99.

Remember that the command will perform the log cycle on whichever node is currently running the specified not application - it might not be the local node!

16.8.1 Handling of Compressed Logs

Once a log file has been rotated and been given a three digit numeric extension it is often useful to compress it to save further space. If this is desired, then any of the following commands are recommended:

- gzip
- compress
- bzip2

When a log file is compressed using these commands then the additional extension added to the file name is automatically taken into account when the files are rotated. For example consider the following log files exist:

```
lems-apache.log.001.gz
lems-apache.log
```

After running the “LOGCYCLE” command the log files visible will be:

```
lems-apache.log.002.gz
lems-apache.log.001
lems-apache.log
```

16.9 Stopping and Starting the Lems Daemon Manually

In almost all circumstances it is not necessary to either stop or start the Lems daemon for a given application. Under normal operation the starting and stopping of the clustered application via the standard commands such as “clrunapp” will also start and stop the appropriate Lems daemon.

However since the daemon can include custom monitors written by administrators it is quite possible that a running Lems daemon may die due to failures in such a monitor. Because of this it is useful to understand how to start and stop the daemon manually - and the impact such actions have on the running application and cluster as a whole.

Hence to stop a running Lems daemon simply use the kill command with the appropriate PID - for example;

```
# kill $(ps -ef | grep lems-samba | grep -v grep|awk '{print $2}')
```

However the best approach is to make use of “lemsctl”:

```
# lemsctl --application samba --msg ABORT
OK
```

Manually starting the lems daemon is also just as straightforward - though the command to type is somewhat lengthy. For example to start Lems for the again using the “Samba” application as an example:

```
# lems.pl --detach --application samba --verbose \  
--file /var/log/cluster/lems/lems-samba.log \  
--config /etc/cluster/samba/lems.local.xml
```

The Lems daemon is critical to operations and so the running cluster daemon on a node will restart it if a previously running Lems daemon appears to die or stop. Thus killing the Lems daemon is a good way to force a restart. To simply stop the monitoring use the “lemsctl” command with the PAUSE option instead.

17 Managing Cluster Daemons with “cldaemonctl”

In almost all cases there is no need to actively “manage” the cluster daemons – the processes run until they are told to stop, suffer from an error, or are killed off. However in certain circumstances it is often helpful if the administrator is aware of various interactions that are possible directly from the command line to manage the environment.

17.1 Log File management

Verbose logging is one feature that although optional, is used in almost all environments. However the cluster daemons can produce a significant amount of output, as this sample “logs” directory indicates:

```
root@serverc:/var/log/cluster# ls -lrt
total 13621
-rw-r--r-- 1 root root 0 Jan 17 09:49 clhalt.apache.log
-rw-r--r-- 1 root root 208 Jan 17 09:49 apache.stop.log
-rw-r--r-- 1 root root 265 Jan 17 09:51 apache.start.log
drwxr-xr-x 2 root bin 144 Mar 15 19:24 lems/
-rw-r--r-- 1 root root 3197 Mar 23 19:55 clhalt.log
-rw-r--r-- 1 root root 0 Mar 23 22:28 mysql.stop.errlog
-rw-r--r-- 1 root root 15849 Mar 23 22:32 clstart.mysql.log
-rw-r--r-- 1 root root 1460 Apr 1 22:48 mysql.stop.log
-rw-r--r-- 1 root root 1601 Apr 1 22:48 mysql.start.log
-rw-r--r-- 1 root root 6221 Apr 1 22:48 clhalt.mysql.log
-rw-r--r-- 1 root root 330 Apr 2 22:57 samba.start.log
-rw-r--r-- 1 root root 99 Apr 2 22:57 samba.start.errlog
-rw-r--r-- 1 root root 32744 Apr 2 23:59 clhalt.samba.log
-rw-r--r-- 1 root root 1276 Apr 3 00:01 samba.stop.errlog
-rw-r--r-- 1 root root 6511 Apr 3 00:19 samba.stop.log
-rw-r--r-- 1 root root 119633 Apr 3 00:23 clstart.samba.log
-rw-r--r-- 1 root root 11713982 Apr 3 14:53 cldaemon-cluster2.log
```

Notice that the log files written by Lems are kept in a separate sub-directory. Management of Lems logs files has been previously discussed on page 118. This directory contains a collection of logs, some related to the cluster overall, others to individual applications. The naming schemes for each log file, and the contents are explained next.

17.1.1 Application Specific Logs

In this directory the administrator will find logs with the following format:

```
<application>.(start|stop).(log|errlog)
```

These are log files generated by the “clstartapp” and “clhaltapp” tools when a specified application starts or stops respectively. The “log” and “errlog” refer to the standard out and standard error messages that are generated during that process.

These files are typically very small since they only capture output that has been missed by the main logging functions. Since these files are used only occasionally the recommended cause of action is to simply delete them once older than a certain period of time (typically 30 days). If application start-ups / shut-downs occur frequently a better approach would be to “tail” them - just keep the most recent 1000 lines for example.

17.1.2 Clstartapp and Clhaltapp specific Logs

Apart from capturing the output of the start-up and shut-down of the applications, the “clstartapp” and “clhaltapp” commands also generate detailed logs (when in verbose mode), describing the various steps and actions they take once invoked. Obviously the size of the logs really depend on how often the applications start and stop, though sizes of up to a 1Mb are not uncommon.

The format of these log files is:

```
(clstart|clhalt).<application>.log
```

Again these files are only written to and kept open during the start-up and shut-down of an application, and hence can be handled via a log rotation tool, (such as Skulker), to ensure that only recent log information is kept. However outright deletion of these files is discouraged – they may contain useful information on error conditions.

17.1.3 Cluster daemon log files

Apart from Lems logs the cluster daemon is likely to produce the largest log files. For a long running cluster it is possible to generate logs of larger than 50 Mb – though the size of logs generation for version 1.0.0 is less than previous versions since most debugging output is only logged optionally. However, as with the Lems daemon logs, this log file can not simply be removed – since it is kept open at all times.

However the utility “cldaemonctl” for which this section is actually for, provides some facilities to handle log file management – in the same manner as was explained for Lems daemons.

The first function that is available is that it is possible to turn the verbosity of the daemon on the local machine on or off. To turn logging off, the administrator would run the following command:

```
# cldaemonctl --msg "VERBOSE off=true"
OK LOGGING=OFF
```

This will turn the logging off for the local cluster daemon only – it would need to be run on both nodes to turn off verbosity on both cluster daemons. A short-cut is available if both daemons are to be changed, namely the use of the “FORWARD” action:

```
# cldaemonctl --msg "VERBOSE on=true forward=yes"
OK LOGGING=ON
```

The above example turns back on logging - for both nodes if both have running cluster daemons.

However even if verbosity is turned off the log file remains open – meaning that it can not be freely managed by a log management tool. However like Lems the cluster daemon has a built in log cycle tool that is used in the same way:

```
# cldaemonctl --msg "LOGCYCLE count=3 forward=yes"
OK
```

The above command will stop logging to the current log and create a log with the following name:

```
cldaemon-<clustertype>.log.000
```

If such a file already exists it is renamed first to “.001”. This process continues to ensure that at most “count” (given in the “cldaemonctl” arguments) log files are kept. Count can be any number from 1 to 99.

17.2 Resetting Application Fail-over Capability

Once a clustered application has suffered from several software re-starts on the same node it will (depending on the “process monitor” configuration in Lems), fail-over the application to the other node in the cluster. Such a fail-over is known as a “software fail-over” - that is it is thought that some software condition on the original server was causing the application to fail. Following such a fail-over the administrator will notice that fail-back to the original node is not possible:

```
# clstat | egrep "App|apache"
Application      Node      State   Started   Monitor   Stale   Fail-over?
  apache         s14s1    STARTED 0:00:03   Running   0       No
```

This indicates that the application has previously failed-over to the other node, and since it was a software fault rather than a hardware failure, the cluster daemons will not allow fail-back to the original node.

The reasoning behind this approach is that until the condition that caused the software failures on the original node are fixed there is little point failing back to this node. These flag conditions are defined per-application so when “samba” has such a software problem other applications (such as “apache” in the sample configuration), will remain unaffected.

If the administrator is happy to allow fail-back to the original node to take place the “cldaemonctl” command can be used to reset the list of nodes that the application considers valid. For example to reset the “apache” application to ensure fail-back to the original node is available, run the following command on either node:

```
# cldaemonctl --msg "SETVALIDNODES app=apache forward=yes"
OK
```

As usual, the “forward=yes” is given to ensure the valid application status is changed on both nodes - this is obviously important!

The responses from the cluster daemon, as typical, are very terse, typically “OK” or another single word to give an indication of the problem. In this case running the “clstat” command again gives:

```
# clstat | egrep "App|apache"
Application      Node      State  Started  Monitor  Stale  Fail-over?
      apache      sl4s1    STARTED 0:00:-50  Running    0      Yes
```

17.3 Stopping Application Fail-over Capability

There are occasions when the administrator does not want the application to software fail-over to the other node. In these situations it is possible for the administrator to indicate that the application should just halt, rather than restart on the other node.

One way of stopping the application from restarting is to simply stop the relevant process monitor. Considering the “samba” application:

Process Monitors

Name	Status	Restarts	Current	Reset at
smbd	Running	1	0	N/A

The monitor could be disabled by running:

```
# lemsctl --application samba --msg "PAUSE smbd"
smbd PAUSED
```

Such operations have been shown and explained in the previous chapter. However an alternative is to allow the process monitor to continue to perform a limited number of local restarts, but not to be able to restart the application on the remote node.

This is possible by using the “SETVALIDNODES” message, which would be used in the following manner:

```
# cldaemonctl --msg "SETVALIDNODES app=samba nodes=sl4s1 forward=yes"
OK
```

Now running “clstat” will indicate that fail-over for that application is not available.

If the “smbd” monitor process was re-started and the application (software) did fail it would be shut-down and **not** started on the other node since that node is not considered valid.

17.4 Checking Cluster Status in a Script

It is often useful to be able to ascertain the status of the cluster from within a shell script. For advanced details of the cluster status the most obvious answer is to parse the output of the "clstat" command, though this is not actually necessary.

The easiest way is to simply do the following:

```
# cldaemonctl --msg ECHO
```

If the cluster is running this will return "UP", and if it is down it will return a string starting with "ERROR".

Please note although it is possible to simply check the for existence of the cluster daemon, using a command such as:

```
if [[ -n "$(ps -ef | awk '$NF == "cldaemon-cluster2"' )" ]]
then
  echo "cluster running"
fi
```

This type of solution is not recommended for two reasons;

- The name of the process is not guaranteed to remain the same for later releases – hence it might cause problems when you the software is upgraded.
- Checking for the existence on a single node is not a guarantee that the cluster is not functioning. If the administrator wishes to check the process table for cluster status, then it should be done on both nodes.

The "cldaemonctl" can be run on either node in the cluster, and it will connect to the local or remote cluster daemon as necessary - and hence the script using it does not need to worry about which nodes are in or out of the cluster.

17.5 Starting and Stopping Applications

Although not the preferred option it is possible to actually start applications using the "cldaemonctl" routine. In this instance the application will get started on whichever node receives the request – which is the local machine if it is part of the cluster, the remote machine otherwise.

To actually start the application simply run the following command:

```
# cldaemonctl --msg "START_APP app=apache"
OK
```

No other arguments are accepted. Since this command is sent directly to a cluster daemon it has some interesting characteristics. It examines the application configuration take-over settings to see if it should exactly do what has been requested. If the takeover setting is "normal" rather than "force" then it will abort if any "STALE" flags exist on this host for this host.

If "STALE" flags do exist, but the takeover setting is "force" they are deleted before attempting to start the application in force mode.

For the current releases these STALE flags are not yet created - though future releases may reintroduce them (probably created via the "fsmonitor" Lems monitor).

In a similar way it is possible to stop the application simply by running:

```
# cldaemonctl --msg "STOP_APP app=apache"
```

However this needs to be run on the node which currently has the application running otherwise the response returned is:

```
NOT_RUNNING
```

The expected response is:

```
OK <pid>
```

The PID returned is the process ID of the “clhaltapp” command that is used to stop the application. This is useful because the request is handled asynchronously – simply expecting the application to have halted when the “cldaemonctl” call returns is not a valid method of waiting for the application to halt. Instead if the specified process ID no longer exists then the application should have stopped.

18 Managing Configured Applications

Over time existing applications typically need to change - and ideally such changes should be able to take place whilst the application in question is still up and running. Linuxha.net is designed to work in exactly this way. The "clbuildapp" can not only be used to build new applications, it can also be used to alter existing ones. Using this utility it is also possible to do the following:

- *Addition of new File systems*
Allows new file systems to be added to an existing application using just a single command - even if the application in question is currently running.
- *Removal of existing File systems*
Remove one or more file systems from an application with just a single command - again even if the application is currently running.
- *Changing existing File systems*
Allows existing file systems to be expanded, or mount options changed. If the file system type in use is supported⁹, it is possible to grow file systems whilst the application is up and running.
- *Modification of application parameters*
If IP addresses need to be added, volume groups changed for example. Currently the changes can be made "online", but the changes will online take affect when the application is restarted.

18.1 Adding new file systems

There are essentially two different ways of adding new file systems to an existing application;

- *On-line* - the application in question is currently up and available for client access.
- *Off-line* - performed when the application is not running in the cluster. The cluster itself, and other applications may or may not be running at the time.

Each will be explained separately. The administrator is free to choose which method to use, the results are equally the same.

18.1.1 On-line Addition of file systems

All the steps in this section **must take place on the node where the application is currently running** (unless stated otherwise). The first thing is to add the new logical volume on the "live" node for the application, for example:

```
# lvcreate -L 50m -n data2 apachevg
```

Once created the administrator may optionally perform the same step on the other node. This is optional - and is only necessary if the administrator wishes to ensure the logical volume is created on certain physical disks in the volume group.

⁹ Currently jfs, reiserfs and xfs are supported. ext3 support is built in to the software but is untested since the kernel changes required to support on-line file system expansion for ext3 are not commonly in use as yet.

At this point it will be necessary to create the file system, file system mount point and mount it, all on the “live” machine:

```
# mkfs -t xfs /dev/apachevg/data2
meta-data=/dev/apachevg/data2    isize=256    agcount=3, agsize=4096 blks
      =                               sectsz=512
data      =                               bsize=4096    blocks=12288, imaxpct=25
      =                               sunit=0      swidth=0 blks, unwritten=1
naming    =version 2                bsize=4096
log       =internal log            bsize=4096    blocks=1200, version=1
      =                               sectsz=512    sunit=0 blks
realtime  =none                    extsz=65536   blocks=0, rtextents=0
# mkdir /apache/data2
# chmod 555 /apache/data2
# mount /dev/apachevg/data2 /apache/data2
# chmod 555 /apache/data2
```

The following points should be remembered:

- The file system is mounted as part of the **local** storage on the node - no replication is currently taking place.
- The mount point and permissions must be replicated on the other node
- The file system does not need to be created or mounted on the other node
- None of the changes so far affect the cluster or application configuration
- **Only add file systems when both nodes are available and are running in the cluster.**

Thus before continuing the administrator should ensure the mount point and permissions are set the same by running the following on the **other node**:

```
# mkdir /apache/data2
# chmod 555 /apache/data2
```

At this point, consider the “df” output on the server where the “apache” application is currently live:

```
# df
Filesystem          1K-blocks      Used Available Use% Mounted on
/dev/hda1            2016016    1803048    110556  95% /
none                 128020         0    128020   0% /dev/shm
/dev/drbd13           11584         48    11536   1% /samba/cfg
/dev/drbd14           11584        460    11124   4% /samba/logs
/dev/drbd12           44352         64    44288   1% /samba/shares
/dev/drbd0            126272        316    125956   1% /apache
/dev/mapper/apachevg-data2
                    44352         64    44288   1% /apache/data2
```

The new file system is using the “/dev/mapper/apachevg-data2” device rather than a “DRBD” device, verifying that it is only locally available storage. It also appears that the “samba” application is running on this node as well at present, though that is of no consequence.

Now the application must be rebuilt. First the “clbuildapp” command is run with the “--vgbuild” option on the live node. The “-F” is necessary since the application is already defined and is actually running.

```
# clbuildapp -A apache -V -F --vgbuild
INFO 27/06/2005 12:33:12 Backups directory defaulted to /clbackup
INFO 27/06/2005 12:33:13 Attempting to backup cluster configuration on node sl4s1.
INFO 27/06/2005 12:33:13 Saved cluster configuration in 38244 bytes on node sl4s1
INFO 27/06/2005 12:33:13 Attempting to backup cluster configuration on node sl4s2.
INFO 27/06/2005 12:33:13 Saved cluster configuration in 11537 bytes on node sl4s2
INFO 27/06/2005 12:33:14 Validated clconf.xml is consistent on both nodes
. . .
INFO 27/06/2005 12:33:15 Preferred_node parameter present and valid.
WARN 27/06/2005 12:33:15 Application rebuild selected for running application.
INFO 27/06/2005 12:33:15 Number of networks to validate: 1
WARN 27/06/2005 12:33:15 Multiple IP addresses detected for network prod.
INFO 27/06/2005 12:33:15 Validated network prod defined in cluster topology.
WARN 27/06/2005 12:33:15 Network prod missing netmask - will use default.
```

```

WARN 27/06/2005 12:33:15 Network prod missing broadcast - will use default.
INFO 27/06/2005 12:33:15 Validated network prod attributes.
INFO 27/06/2005 12:33:15 Status directory on sl4s1 already exists.
INFO 27/06/2005 12:33:15 Validated/created status directory on sl4s2
INFO 27/06/2005 12:33:15 REBUILD: Original fsmap entry count: 1
INFO 27/06/2005 12:33:15 Getting list of defined volume groups on sl4s2
INFO 27/06/2005 12:33:16 VG apachevg does not contain LV data2_meta on sl4s1 -
creating...
INFO 27/06/2005 12:33:16 VG/LV apachevg/data2_meta of 131072Kb built on sl4s1
INFO 27/06/2005 12:33:16 Validated Meta volume exists for lv01 (and is 128mb)
INFO 27/06/2005 12:33:16 VG apachevg does not contain LV data2 on sl4s2 - creating...
INFO 27/06/2005 12:33:17 VG/LV apachevg/data2 of 53248Kb built on sl4s2
INFO 27/06/2005 12:33:18 VG apachevg does not contain LV data2_meta on sl4s2 -
creating...
INFO 27/06/2005 12:33:18 VG/LV apachevg/data2_meta of 131072Kb built on sl4s2
INFO 27/06/2005 12:33:19 All LV checks on VG apachevg completed
INFO 27/06/2005 12:33:19
INFO 27/06/2005 12:33:19 Now run with the --build to ensure all required resources
INFO 27/06/2005 12:33:19 are allocated on both nodes
INFO 27/06/2005 12:33:19

```

followed by:

```

# clbuildapp -A apache -V -F --build
. . .
WARN 27/06/2005 12:42:24 Application rebuild selected for running application.
INFO 27/06/2005 12:42:24 Number of networks to validate: 1
WARN 27/06/2005 12:42:24 Multiple IP addresses detected for network prod.
INFO 27/06/2005 12:42:24 Validated network prod defined in cluster topology.
WARN 27/06/2005 12:42:24 Network prod missing netmask - will use default.
WARN 27/06/2005 12:42:24 Network prod missing broadcast - will use default.
INFO 27/06/2005 12:42:24 Validated network prod attributes.
INFO 27/06/2005 12:42:24 Status directory on sl4s1 already exists.
INFO 27/06/2005 12:42:25 Validated/created status directory on sl4s2
INFO 27/06/2005 12:42:25 Validated VG build run has completed against this
configuration.
INFO 27/06/2005 12:42:25 REBUILD: Original fsmap entry count: 1
INFO 27/06/2005 12:42:25 LVM on sl4s1 appears to be version 2
INFO 27/06/2005 12:42:25 LVM on sl4s2 appears to be version 2
INFO 27/06/2005 12:42:25 Checked volume groups exist on sl4s1
INFO 27/06/2005 12:42:25 Checked volume groups exist on sl4s2
INFO 27/06/2005 12:42:25 checking if apachevg/data2 open...
INFO 27/06/2005 12:42:25 checking if apachevg/lv01 mounted as /apache...
INFO 27/06/2005 12:42:25 Checked 4 logical volumes for apachevg on sl4s1
INFO 27/06/2005 12:42:25 Completed volume group analysis on sl4s1:
INFO 27/06/2005 12:42:25   VG: apachevg   Used LVs: 4 Not Used LVs: 0
INFO 27/06/2005 12:42:25 sl4s1 available: PORTS: 95, DRBD: 43
INFO 27/06/2005 12:42:26 sl4s2 available: PORTS: 95, DRBD: 43
INFO 27/06/2005 12:42:26 REBUILD: Validated enough resources available on both nodes
INFO 27/06/2005 12:42:26 REBUILD: Ports/DRBD resources required: 1
INFO 27/06/2005 12:42:26 Host/VG/LV sl4s1/apachevg/data2 allocated port 9905
INFO 27/06/2005 12:42:27 Host/VG/LV sl4s2/apachevg/data2 allocated port 9905
INFO 27/06/2005 12:42:27 Host/VG/LV sl4s1/apachevg/data2 allocated DRBD 15
INFO 27/06/2005 12:42:28 Host/VG/LV sl4s2/apachevg/data2 allocated DRBD 15
INFO 27/06/2005 12:42:28 Host/VG/LV sl4s1/apachevg/lv01 allocated port 9901
INFO 27/06/2005 12:42:28 Host/VG/LV sl4s2/apachevg/lv01 allocated port 9901
INFO 27/06/2005 12:42:28 Host/VG/LV sl4s1/apachevg/lv01 allocated DRBD 0
INFO 27/06/2005 12:42:29 Host/VG/LV sl4s2/apachevg/lv01 allocated DRBD 0
INFO 27/06/2005 12:42:29 Validated/created application status directory on sl4s2
INFO 27/06/2005 12:42:29 REBUILD: Current fsmap being replaced on sl4s1.
INFO 27/06/2005 12:42:29 Validated/created fsmap directory on sl4s1
INFO 27/06/2005 12:42:29 Validated/created fsmap directory on sl4s2
INFO 27/06/2005 12:42:30 Validated/created application config directory on sl4s2
INFO 27/06/2005 12:42:30 Copied config data for apache to sl4s2
INFO 27/06/2005 12:42:30 REBUILD: Getting details for /samba/cfg from old fsmap...
WARN 27/06/2005 12:42:30 Unable to get information for /samba/cfg ... ignoring!!
INFO 27/06/2005 12:42:30 REBUILD: Getting details for /samba/logs from old fsmap...
WARN 27/06/2005 12:42:30 Unable to get information for /samba/logs ... ignoring!!
INFO 27/06/2005 12:42:30 REBUILD: Getting details for /samba/shares from old fsmap...
WARN 27/06/2005 12:42:30 Unable to get information for /samba/shares ... ignoring!!
INFO 27/06/2005 12:42:30 REBUILD: Getting details for /apache from old fsmap...
INFO 27/06/2005 12:42:30 REBUILD: Written apachevg:lv01:/apache:xf:rw:131072
INFO 27/06/2005 12:42:31 Validated file system mount point /apache/data2
INFO 27/06/2005 12:42:31 REBUILD: Removing original fsmap file on sl4s1.
INFO 27/06/2005 12:42:31 REBUILD: Recreated fsmap data for apache on sl4s1

```



```

INFO 27/06/2005 12:42:32 Copied fsmap data for apache to sl4s2
INFO 27/06/2005 12:42:32 REBUILD: Un-mounted local /apache/data2.
INFO 27/06/2005 12:42:33 REBUILD: DRBD device for /apache/data2 started to
synchronise.
INFO 27/06/2005 12:42:33 REBUILD: Fsck for /apache/data2; "/sbin/fsck -t xfs -a
/dev/drbd15" ...
INFO 27/06/2005 12:42:33 REBUILD: Mounting file system /apache/data2
(PATH=$PATH:/sbin:/bin:/usr/sbin; mount -t xfs -o rw /dev/drbd15 /apache/data2)
INFO 27/06/2005 12:42:34 REBUILD: File system /apache/data2 mounted successfully.
INFO 27/06/2005 12:42:34 Application apache now (re)registered with cluster.
INFO 27/06/2005 12:42:34 Lems response to reconfigure: OK
INFO 27/06/2005 12:42:34 Transferring XML file lems.local.xml to sl4s2.
INFO 27/06/2005 12:42:34 Transferring XML file httpd.xml to sl4s2.
INFO 27/06/2005 12:42:35
INFO 27/06/2005 12:42:35 Cluster build has completed successfully.
INFO 27/06/2005 12:42:35 The rebuild addition / removal of file systems has been
INFO 27/06/2005 12:42:35 completed without problems. Please note that the new file
systems
INFO 27/06/2005 12:42:35 may still be in the process of being synchronised.
INFO 27/06/2005 12:42:35

```

There is no requirement to explicitly attempt to synchronise the new storage - that will be done automatically.

In the above output messages checking file systems in other volume groups, ("samba" in this case) are shown - that will be resolved for version 1.0.0 onwards.

At this point the file system will have been remounted to work using a DRBD device, and so can be readily used by the application from this point onwards. Validate this fact using "df":

```

# df
Filesystem          1K-blocks      Used Available Use% Mounted on
/dev/hdal            2016016    1803204    110400   95% /
none                128020         0    128020    0% /dev/shm
/dev/drbd13         11584         48    11536    1% /samba/cfg
/dev/drbd14         11584        460    11124    4% /samba/logs
/dev/drbd12         44352         64    44288    1% /samba/shares
/dev/drbd0          126272        316   125956    1% /apache
/dev/drbd15         44352         64    44288    1% /apache/data2

```

The "fsmonitor" Lems sub-system is reloaded for the application thus ensuring data synchronisation to start automatically for the new file system after a short period of time. Again use the "clstat" to validate this fact:

```

# clstat -A apache
Cluster: sl4cluster - UP

Application      Node      State  Runnig  Monitor  Stale  Fail-over?
  apache        sl4s1    STARTED  0:00:15  Running    1      Yes

File Systems

Mount Point      Valid  Type    State    % Complete  Completion
/apache          both   drbd    Sync
/apache/data2    local  drbd    Syncing    29 %    0:04:12:00

Process Monitors

Name    Status  Restarts  Current  Reset at
httpd   Running    3         0        N/A

General Monitors

Type      Name      Status
Flag Check  flag_check  Running
FS Monitor  fsmonitor  Running
IP Monitor  ipcheck    Running

```

At this point the file system is ready to use and no further work will be necessary.

18.1.2 Off-line Addition of file systems

Off-line changes are more difficult! This is due to the fact that the references to existing and current configuration that an on-line change can make use of are not present. Thus off-line changes are discouraged if possible.

To make an off-line change please ensure the following **all commands are run on the "primary" node** (unless stated otherwise) - that is the node that is defined first in the cluster configuration.

Firstly the administrator must manually mount **ALL** the file systems that are being used as part of the application, using the local storage devices - even those that are currently defined:

```
# mount /dev/apachevg/lv01 /apache
```

At this point the new logical volume should be created on the local machine, the file system created and then mounted at the required location:

```
# lvcreate -L 1g -n data2 vg02
```

Once created the administrator may optionally perform the same step on the other node. This is optional - and is only necessary if the administrator wishes to ensure the logical volume is created on certain physical disks in the volume group.

Continue with the necessary commands:

```
# mkfs -t xfs /dev/apachevg/data2
# mkdir /apache/data2
# chmod 555 /apache/data2
# mount /dev/apachevg/data2 /apache/data2
# chmod 555 /apache/data2
```

Ensure the same directory path and permissions are available on the other node:

```
# mkdir /apache/data2
# chmod 555 /apache/data2
```

Now the administrator can run the commands necessary to perform the off-line rebuild:

```
# clbuildapp -A apache -V -F --vgbuild
INFO 27/06/2005 13:05:45 Current status of application apache is: DOWN
INFO 27/06/2005 13:05:45 Number of networks to validate: 1
WARN 27/06/2005 13:05:45 Multiple IP addresses detected for network prod.
INFO 27/06/2005 13:05:45 Validated network prod defined in cluster topology.
WARN 27/06/2005 13:05:45 Network prod missing netmask - will use default.
WARN 27/06/2005 13:05:45 Network prod missing broadcast - will use default.
INFO 27/06/2005 13:05:45 Validated network prod attributes.
INFO 27/06/2005 13:05:45 Status directory on s14s1 already exists.
INFO 27/06/2005 13:05:45 Validated/created status directory on s14s2
INFO 27/06/2005 13:05:45 Getting list of defined volume groups on s14s2
INFO 27/06/2005 13:05:46 VG apachevg does not contain LV data2_meta on s14s1 -
creating...
INFO 27/06/2005 13:05:46 VG/LV apachevg/data2_meta of 131072Kb built on s14s1
INFO 27/06/2005 13:05:46 Validated Meta volume exists for lv01 (and is 128mb)
INFO 27/06/2005 13:05:46 VG apachevg does not contain LV data2 on s14s2 - creating...
INFO 27/06/2005 13:05:47 VG/LV apachevg/data2 of 53248Kb built on s14s2
INFO 27/06/2005 13:05:48 VG apachevg does not contain LV data2_meta on s14s2 -
creating...
INFO 27/06/2005 13:05:48 VG/LV apachevg/data2_meta of 131072Kb built on s14s2
INFO 27/06/2005 13:05:48 All LV checks on VG apachevg completed
INFO 27/06/2005 13:05:49
INFO 27/06/2005 13:05:49 Now run with the --build to ensure all required resources
INFO 27/06/2005 13:05:49 are allocated on both nodes
INFO 27/06/2005 13:05:49
```

Followed by:

```
# clbuildapp -A apache -V -F --build --fsmap
INFO 27/06/2005 13:14:43 Current status of application apache is: DOWN
INFO 27/06/2005 13:14:43 Number of networks to validate: 1
WARN 27/06/2005 13:14:43 Multiple IP addresses detected for network prod.
INFO 27/06/2005 13:14:43 Validated network prod defined in cluster topology.
WARN 27/06/2005 13:14:43 Network prod missing netmask - will use default.
WARN 27/06/2005 13:14:43 Network prod missing broadcast - will use default.
INFO 27/06/2005 13:14:43 Validated network prod attributes.
INFO 27/06/2005 13:14:43 Status directory on sl4s1 already exists.
INFO 27/06/2005 13:14:43 Validated/created status directory on sl4s2
INFO 27/06/2005 13:14:43 Validated VG build run has completed against this
configuration.
INFO 27/06/2005 13:14:43 REBUILD: Original fsmap entry count: 1
INFO 27/06/2005 13:14:43 LVM on sl4s1 appears to be version 2
INFO 27/06/2005 13:14:44 LVM on sl4s2 appears to be version 2
INFO 27/06/2005 13:14:44 Checked volume groups exist on sl4s1
INFO 27/06/2005 13:14:44 Checked volume groups exist on sl4s2
WARN 27/06/2005 13:14:44 Running --build for an existing application, whilst
WARN 27/06/2005 13:14:44 that application is not running will not update
WARN 27/06/2005 13:14:44 the file system mappings. If this is required for
WARN 27/06/2005 13:14:44 an existing application please start it first.
INFO 27/06/2005 13:14:44 checking if apachevg/data2 open...
INFO 27/06/2005 13:14:44 Checked 4 logical volumes for apachevg on sl4s1
INFO 27/06/2005 13:14:44 Completed volume group analysis on sl4s1:
INFO 27/06/2005 13:14:44     VG: apachevg  Used LVs: 4 Not Used LVs: 0
INFO 27/06/2005 13:14:44 sl4s1 available: PORTS: 95, DRBD: 43
INFO 27/06/2005 13:14:44 sl4s2 available: PORTS: 95, DRBD: 43
INFO 27/06/2005 13:14:44 REBUILD: Validated enough resources available on both nodes
INFO 27/06/2005 13:14:44 REBUILD: Ports/DRBD resources required: 1
INFO 27/06/2005 13:14:45 Host/VG/LV sl4s1/apachevg/data2 allocated port 9905
INFO 27/06/2005 13:14:46 Host/VG/LV sl4s2/apachevg/data2 allocated port 9905
INFO 27/06/2005 13:14:46 Host/VG/LV sl4s1/apachevg/data2 allocated DRBD 15
INFO 27/06/2005 13:14:46 Host/VG/LV sl4s2/apachevg/data2 allocated DRBD 15
INFO 27/06/2005 13:14:46 Host/VG/LV sl4s1/apachevg/lv01 allocated port 9901
INFO 27/06/2005 13:14:47 Host/VG/LV sl4s2/apachevg/lv01 allocated port 9901
INFO 27/06/2005 13:14:47 Host/VG/LV sl4s1/apachevg/lv01 allocated DRBD 0
INFO 27/06/2005 13:14:47 Host/VG/LV sl4s2/apachevg/lv01 allocated DRBD 0
INFO 27/06/2005 13:14:48 Validated/created application status directory on sl4s2
INFO 27/06/2005 13:14:48 REBUILD: Current fsmap being replaced on sl4s1.
INFO 27/06/2005 13:14:48 Validated/created fsmap directory on sl4s1
INFO 27/06/2005 13:14:48 Validated/created fsmap directory on sl4s2
INFO 27/06/2005 13:14:48 Validated/created application config directory on sl4s2
INFO 27/06/2005 13:14:48 Copied config data for apache to sl4s2
INFO 27/06/2005 13:14:48 REBUILD: Removing original fsmap file on sl4s1.
INFO 27/06/2005 13:14:48 REBUILD: Recreated fsmap data for apache on sl4s1
INFO 27/06/2005 13:14:49 Copied fsmap data for apache to sl4s2
INFO 27/06/2005 13:14:49 Application apache now (re)registered with cluster.
INFO 27/06/2005 13:14:49 Transferring XML file lems.local.xml to sl4s2.
INFO 27/06/2005 13:14:49 Transferring XML file httpd.xml to sl4s2.
INFO 27/06/2005 13:14:50
INFO 27/06/2005 13:14:50 Cluster build has completed successfully.
INFO 27/06/2005 13:14:50 The rebuild addition / removal of file systems has been
INFO 27/06/2005 13:14:50 completed without problems. Please note that the new file
systems
INFO 27/06/2005 13:14:50 may still be in the process of being synchronised.
INFO 27/06/2005 13:14:50
```

The "--fsmap" flag is very important in the above command.

If it is was **not** used for the "--build" stage, the following command would need to be run on each node in order to update the "fsmap" file manually.

```
# echo "apachevg:data2:/apache/data2:xf:rw:$((52*1024))" \
  >>/etc/cluster/.resources/fsmap/apache
```

Unlike the on-line build it is necessary to include an explicit synchronisation;

```
# clbuildapp -A apache -V -F --sync
. . .
INFO 27/06/2005 13:24:16 Loading details of file systems to unmount
INFO 27/06/2005 13:24:16 File system /apache/data2 un-mounted
INFO 27/06/2005 13:24:16 File system /apache un-mounted
INFO 27/06/2005 13:24:17 All file systems un-mounted successfully.
INFO 27/06/2005 13:24:17 Some/all drbd devices need configuration on sl4s1
INFO 27/06/2005 13:24:18 All drbd services now running on sl4s1
INFO 27/06/2005 13:24:19 Attempted to start new drbd services on sl4s2
INFO 27/06/2005 13:24:20 All drbd services now running on sl4s2
INFO 27/06/2005 13:24:21 Successfully started rebuild of DRBD devices for apache.
INFO 27/06/2005 13:24:21 Sync status: Devices sync'ed: 1, Devices un-sync'ed: 0
INFO 27/06/2005 13:24:21 Sync status: Currently syncing LV /dev/apachevg/data2, 0.8%
completed
INFO 27/06/2005 13:24:26 Sync status: Devices sync'ed: 1, Devices un-sync'ed: 0
INFO 27/06/2005 13:24:26 Sync status: Currently syncing LV /dev/apachevg/data2, 19.2%
completed
INFO 27/06/2005 13:24:31 Sync status: Devices sync'ed: 1, Devices un-sync'ed: 0
INFO 27/06/2005 13:24:31 Sync status: Currently syncing LV /dev/apachevg/data2, 37.5%
completed
INFO 27/06/2005 13:24:36 Sync status: Devices sync'ed: 1, Devices un-sync'ed: 0
INFO 27/06/2005 13:24:36 Sync status: Currently syncing LV /dev/apachevg/data2, 56.3%
completed
INFO 27/06/2005 13:24:41 Sync status: Devices sync'ed: 1, Devices un-sync'ed: 0
INFO 27/06/2005 13:24:41 Sync status: Currently syncing LV /dev/apachevg/data2, 74.7%
completed
INFO 27/06/2005 13:24:46 Sync status: Devices sync'ed: 1, Devices un-sync'ed: 0
INFO 27/06/2005 13:24:46 Sync status: Currently syncing LV /dev/apachevg/data2, 93.4%
completed
INFO 27/06/2005 13:24:51 Sync status: Devices sync'ed: 2, Devices un-sync'ed: 0
INFO 27/06/2005 13:24:51 All logical volumes on remote host sl4s2 synchronised
INFO 27/06/2005 13:24:51 Stopping DRBD on sl4s1 devices.
INFO 27/06/2005 13:24:52 Stopping DRBD on sl4s2 devices.
INFO 27/06/2005 13:24:53
INFO 27/06/2005 13:24:53 Successfully Synchronised data from sl4s1 to sl4s2
INFO 27/06/2005 13:24:53
```

Once the standard “--sync” option has been run, the following command MUST ALSO BE RUN:

```
# clbuildapp -A apache -V -F --sync --forcesync
```

The synchronisation steps are absolutely necessary here, since all file systems were mounted locally. Failure to do both steps will result in data corruption across all file systems for the specified application.

Because of the above warning it is reiterated that off-line rebuilds of applications are discouraged!

18.2 Removing existing file systems

In a similar way to adding new file systems, existing file systems can be removed in basically the same two ways;

- *On-line* - Whilst the application is up and running file systems can be removed without disturbing existing client access.
- *Off-line* - Reconfigure the application environment whilst it is not running. The cluster and other clustered application may or may not be running.

18.2.1 On-line Removal of File systems

On the node running the application simply un-mount any file systems that are no longer required, for example:

```
# umount /apache/data2
```

Any processes that make use of the file system will obviously need to be stopped and reconfigured if necessary. If the application is stopped the administrator is reminded to consider using the "lmsctl -A app --msg PAUSE" command beforehand.

Once un-mounted run the following commands on the node where the application currently runs:

```
# clbuildapp -A apache -F -V --vgbuild
. . .
WARN 27/06/2005 12:57:31 Application rebuild selected for running application.
INFO 27/06/2005 12:57:31 Number of networks to validate: 1
WARN 27/06/2005 12:57:31 Multiple IP addresses detected for network prod.
INFO 27/06/2005 12:57:31 Validated network prod defined in cluster topology.
WARN 27/06/2005 12:57:31 Network prod missing netmask - will use default.
WARN 27/06/2005 12:57:31 Network prod missing broadcast - will use default.
INFO 27/06/2005 12:57:31 Validated network prod attributes.
INFO 27/06/2005 12:57:31 Status directory on sl4s1 already exists.
INFO 27/06/2005 12:57:32 Validated/created status directory on sl4s2
INFO 27/06/2005 12:57:32 REBUILD: Original fsmap entry count: 2
INFO 27/06/2005 12:57:32 Getting list of defined volume groups on sl4s2
INFO 27/06/2005 12:57:32 Validated Meta volume exists for data2 (and is 128mb)
INFO 27/06/2005 12:57:32 Validated Meta volume exists for lv01 (and is 128mb)
INFO 27/06/2005 12:57:33 All LV checks on VG apachevg completed
INFO 27/06/2005 12:57:34
INFO 27/06/2005 12:57:34 Now run with the --build to ensure all required resources
INFO 27/06/2005 12:57:34 are allocated on both nodes
INFO 27/06/2005 12:57:34
```

This should be followed by:

```
# clbuildapp -A apache -F -V --build
. . .
WARN 27/06/2005 12:57:57 Application rebuild selected for running application.
WARN 27/06/2005 12:57:57 *** BETA FEATURE ***
INFO 27/06/2005 12:57:57 Number of networks to validate: 1
WARN 27/06/2005 12:57:57 Multiple IP addresses detected for network prod.
INFO 27/06/2005 12:57:57 Validated network prod defined in cluster topology.
WARN 27/06/2005 12:57:57 Network prod missing netmask - will use default.
WARN 27/06/2005 12:57:57 Network prod missing broadcast - will use default.
INFO 27/06/2005 12:57:57 Validated network prod attributes.
INFO 27/06/2005 12:57:57 Status directory on sl4s1 already exists.
INFO 27/06/2005 12:57:57 Validated/created status directory on sl4s2
INFO 27/06/2005 12:57:57 Validated VG build run has completed against this
configuration.
INFO 27/06/2005 12:57:57 REBUILD: Original fsmap entry count: 2
INFO 27/06/2005 12:57:57 LVM on sl4s1 appears to be version 2
INFO 27/06/2005 12:57:57 LVM on sl4s2 appears to be version 2
INFO 27/06/2005 12:57:57 Checked volume groups exist on sl4s1
INFO 27/06/2005 12:57:58 Checked volume groups exist on sl4s2
INFO 27/06/2005 12:57:58 checking if apachevg/data2 mounted as /apache/data2...
INFO 27/06/2005 12:57:58 checking if apachevg/lv01 mounted as /apache...
INFO 27/06/2005 12:57:58 Checked 4 logical volumes for apachevg on sl4s1
INFO 27/06/2005 12:57:58 Completed volume group analysis on sl4s1:
INFO 27/06/2005 12:57:58 VG: apachevg Used LVs: 3 Not Used LVs: 1
INFO 27/06/2005 12:57:58 sl4s1 available: PORTS: 94, DRBD: 42
INFO 27/06/2005 12:57:58 sl4s2 available: PORTS: 94, DRBD: 42
INFO 27/06/2005 12:57:58 REBUILD: No new resources on sl4s1 required.
INFO 27/06/2005 12:58:00 REBUILD: Removed port resource 9905 for apachevg/data2 on
host sl4s1.
INFO 27/06/2005 12:58:00 REBUILD: Removed DRBD resource 15 for apachevg/data2 on host
sl4s1.
INFO 27/06/2005 12:58:00 REBUILD: Removed port resource 9905 for apachevg/data2 on
host sl4s2.
INFO 27/06/2005 12:58:01 REBUILD: Removed DRBD resource 15 for apachevg/data2 on host
sl4s2.
INFO 27/06/2005 12:58:01 Host/VG/LV sl4s1/apachevg/lv01 allocated port 9901
INFO 27/06/2005 12:58:01 Host/VG/LV sl4s2/apachevg/lv01 allocated port 9901
INFO 27/06/2005 12:58:01 Host/VG/LV sl4s1/apachevg/lv01 allocated DRBD 0
INFO 27/06/2005 12:58:02 Host/VG/LV sl4s2/apachevg/lv01 allocated DRBD 0
INFO 27/06/2005 12:58:02 Validated/created application status directory on sl4s2
INFO 27/06/2005 12:58:02 REBUILD: Current fsmap being replaced on sl4s1.
INFO 27/06/2005 12:58:02 Validated/created fsmap directory on sl4s1
INFO 27/06/2005 12:58:02 Validated/created fsmap directory on sl4s2
```

```

INFO 27/06/2005 12:58:02 Validated/created application config directory on sl4s2
INFO 27/06/2005 12:58:03 Copied config data for apache to sl4s2
INFO 27/06/2005 12:58:03 REBUILD: Getting details for /samba/cfg from old fsmap...
WARN 27/06/2005 12:58:03 Unable to get information for /samba/cfg ... ignoring!!
INFO 27/06/2005 12:58:03 REBUILD: Getting details for /samba/logs from old fsmap...
WARN 27/06/2005 12:58:03 Unable to get information for /samba/logs ... ignoring!!
INFO 27/06/2005 12:58:03 REBUILD: Getting details for /samba/shares from old fsmap...
WARN 27/06/2005 12:58:03 Unable to get information for /samba/shares ... ignoring!!
INFO 27/06/2005 12:58:03 REBUILD: Getting details for /apache from old fsmap...
INFO 27/06/2005 12:58:03 REBUILD: Written apachevg:lv01:/apache:xfs:rw:131072
INFO 27/06/2005 12:58:03 REBUILD: Removing original fsmap file on sl4s1.
INFO 27/06/2005 12:58:03 REBUILD: Recreated fsmap data for apache on sl4s1
INFO 27/06/2005 12:58:03 Copied fsmap data for apache to sl4s2
INFO 27/06/2005 12:58:03 REBUILD: No new file systems currently mounted.
INFO 27/06/2005 12:58:03 Application apache now (re)registered with cluster.
INFO 27/06/2005 12:58:03 Lems response to reconfigure: OK
INFO 27/06/2005 12:58:03 Transferring XML file lems.local.xml to sl4s2.
INFO 27/06/2005 12:58:04 Transferring XML file httpd.xml to sl4s2.
INFO 27/06/2005 12:58:04
INFO 27/06/2005 12:58:04 Cluster build has completed successfully.
INFO 27/06/2005 12:58:04 The rebuild addition / removal of file systems has been
INFO 27/06/2005 12:58:04 completed without problems. Please note that the new file
systems
INFO 27/06/2005 12:58:04 may still be in the process of being synchronised.
INFO 27/06/2005 12:58:04

```

This process will also ensure that the relevant file system entries that are no longer mounted and hence removed are also removed from the "fsmap" file - that file which defines which file systems to mount/un-mount when the application starts or stops.

After the rebuild the administrator can either leave the logical volume or remove it. For example to remove it (and the associated meta-data logical volume), run the following on both hosts:

```

# lvremove -f /dev/vg02/data2
# lvremove -f /dev/vg02/data2_meta

```

18.2.2 Off-line Removal of File systems

As with the off-line addition of file systems the steps here require the administrator to take more care, virtually following the same process. Hence again, such off-line file system removals are discouraged - the on-line method is easier and far less error prone.

Firstly the administrator must manually mount the file systems that are being used as part of the application, excluding any file systems they wish to remove from the application definition;

```

# mount /dev/vg02/data /apache

```

Now the administrator can run the commands necessary to perform the off-line rebuild:

```

# clbuildapp -A apache -V -F --vgbuild
INFO 29/06/2005 15:14:28 Current status of application apache is: DOWN
INFO 29/06/2005 15:14:28 Number of networks to validate: 1
WARN 29/06/2005 15:14:28 Multiple IP addresses detected for network prod.
INFO 29/06/2005 15:14:28 Validated network prod defined in cluster topology.
WARN 29/06/2005 15:14:28 Network prod missing netmask - will use default.
WARN 29/06/2005 15:14:28 Network prod missing broadcast - will use default.
INFO 29/06/2005 15:14:28 Validated network prod attributes.
INFO 29/06/2005 15:14:28 Status directory on sl4s1 already exists.
INFO 29/06/2005 15:14:28 Validated/created status directory on sl4s2
INFO 29/06/2005 15:14:28 Getting list of defined volume groups on sl4s2
INFO 29/06/2005 15:14:30 Validated Meta volume exists for data2 (and is 128mb)
INFO 29/06/2005 15:14:30 Validated Meta volume exists for lv01 (and is 128mb)
INFO 29/06/2005 15:14:32 All LV checks on VG apachevg completed
INFO 29/06/2005 15:14:32
INFO 29/06/2005 15:14:32 Now run with the --build to ensure all required resources
INFO 29/06/2005 15:14:32 are allocated on both nodes
INFO 29/06/2005 15:14:32

```

Followed by:

```
# clbuildapp -A apache -V -F --build
. . .
INFO 29/06/2005 15:15:22 REBUILD: Original fsmap entry count: 2
INFO 29/06/2005 15:15:22 LVM on sl4s1 appears to be version 2
INFO 29/06/2005 15:15:23 LVM on sl4s2 appears to be version 2
INFO 29/06/2005 15:15:23 Checked volume groups exist on sl4s1
INFO 29/06/2005 15:15:23 Checked volume groups exist on sl4s2
INFO 29/06/2005 15:15:23 checking if apachevg/data2 mounted as /apache/data2...
INFO 29/06/2005 15:15:23 checking if apachevg/lv01 mounted as /apache...
INFO 29/06/2005 15:15:23 Checked 4 logical volumes for apachevg on sl4s1
INFO 29/06/2005 15:15:23 Completed volume group analysis on sl4s1:
INFO 29/06/2005 15:15:23 VG: apachevg Used LVs: 3 Not Used LVs: 1
INFO 29/06/2005 15:15:23 sl4s1 available: PORTS: 94, DRBD: 42
INFO 29/06/2005 15:15:24 sl4s2 available: PORTS: 94, DRBD: 42
INFO 29/06/2005 15:15:24 REBUILD: No new resources on sl4s1 required.
INFO 29/06/2005 15:15:26 REBUILD: Removed port resource 9905 for apachevg/data2 on
host sl4s1.
INFO 29/06/2005 15:15:26 REBUILD: Removed DRBD resource 15 for apachevg/data2 on host
sl4s1.
INFO 29/06/2005 15:15:27 REBUILD: Removed port resource 9905 for apachevg/data2 on
host sl4s2.
INFO 29/06/2005 15:15:27 REBUILD: Removed DRBD resource 15 for apachevg/data2 on host
sl4s2.
INFO 29/06/2005 15:15:27 Host/VG/LV sl4s1/apachevg/lv01 allocated port 9901
INFO 29/06/2005 15:15:28 Host/VG/LV sl4s2/apachevg/lv01 allocated port 9901
INFO 29/06/2005 15:15:28 Host/VG/LV sl4s1/apachevg/lv01 allocated DRBD 0
INFO 29/06/2005 15:15:28 Host/VG/LV sl4s2/apachevg/lv01 allocated DRBD 0
INFO 29/06/2005 15:15:29 Validated/created application status directory on sl4s2
INFO 29/06/2005 15:15:29 REBUILD: Current fsmap being replaced on sl4s1.
INFO 29/06/2005 15:15:29 Validated/created fsmap directory on sl4s1
INFO 29/06/2005 15:15:29 Validated/created fsmap directory on sl4s2
INFO 29/06/2005 15:15:29 Validated/created application config directory on sl4s2
INFO 29/06/2005 15:15:30 Copied config data for apache to sl4s2
INFO 29/06/2005 15:15:31 Validated file system mount point /apache
INFO 29/06/2005 15:15:31 REBUILD: Removing original fsmap file on sl4s1.
INFO 29/06/2005 15:15:31 REBUILD: Recreated fsmap data for apache on sl4s1
INFO 29/06/2005 15:15:31 Copied fsmap data for apache to sl4s2
INFO 29/06/2005 15:15:32 Application apache now (re)registered with cluster.
INFO 29/06/2005 15:15:32 Transferring XML file lems.local.xml to sl4s2.
INFO 29/06/2005 15:15:32 Transferring XML file httpd.xml to sl4s2.
INFO 29/06/2005 15:15:33
INFO 29/06/2005 15:15:33 Cluster build has completed successfully.
INFO 29/06/2005 15:15:33 The rebuild addition / removal of file systems has been
INFO 29/06/2005 15:15:33 completed without problems. Please note that the new file
systems
INFO 29/06/2005 15:15:33 may still be in the process of being synchronised.
INFO 29/06/2005 15:15:33
```

The "--fsmap" is important here since it cases the existing "fsmap" file to be re-built.

Just as with the off-line addition of file systems it is necessary to include an explicit synchronisation;

```
# clbuildapp -A apache -V -F --sync --forcesync
```

The forced synchronisation step is absolutely necessary here, since all remaining file systems were mounted locally. However due to DRBD this synchronisation will appear almost instantaneous since very little data will actually require replicating to the other node in the cluster.

18.3 Changing existing file systems

Linuxha.net now supports the ability to modify the size of existing file systems. As with the addition and removal this can be done on-line or off-line. The facility to grow an existing file system on-line is determined by the file system, and currently the following have been tested as working;

➤ reiserfs

- xfs
- jfs

Currently “ext3” support is included but remains untested due the lack of availability of the ext3 on-line resize support. Since the size of the device is part of the meta-data kept for DRBD, currently it is recommended that the “on-line” method below is followed - it is certainly far more straightforward.

18.3.1 On-line file system expansion

Unless otherwise specified **all commands documented should be run on the node where the application is currently running**. Also the administrator, as with all changes references in this section, should only attempt to make the changes when both nodes are operating normally.

The first step to perform is to change the underlying size of the local logical volume, for example:

```
# lvextend -L 2g /dev/vg02/apache
```

The same action can optionally be performed on the other node in the cluster. If this is not done the rebuild script will make the change, though of course in this instance the administrator has no control on which extents will be utilised for the new storage.

Once complete run the following commands to ensure the DRBD device size is increased as required:

```
# clbuildapp -A apache -V -F --vgbuild
<output here>
```

Follow this by:

```
# clbuildapp -A apache -V -F --build
<output here>
```

As can be seen from the output the process of growing the file system can take a little while, the larger the logical volume the longer it will take. Once the logical volume change has been handled via the DRBD device the file system is automatically grown.

No other steps are necessary. The new space can be used immediately following the completion of the “clbuildapp” commands shown.

18.3.2 Off-line file system expansion

This is quite a complex operation due to the fact that the DRBD device meta data must be updated. Hence the following warning:

Failure to perform all steps here could result in file system corruption.

On the primary node start the DRBD devices for the application:

```
# /sbin/cluster/utils/drbd_tool --action=start --noprimary \
  --application apache
```

On the other node also start the DRBD devices:

```
# /sbin/cluster/utils/drbd_tool --action=start --noprimary \
  --application apache
```

On the primary node indicate local devices are “primary”:

```
# /sbin/cluster/utils/drbd_tool --action=set_primary --application apache
```


At this point the logical volumes can be grown. Since the offline process is manual, the administrator must ensure the changes can place on both nodes;

```
# lvextend -L 2g /dev/vg02/apache
# ssh server2 lvextend -L 2g /dev/vg02/apache
```

At this point the following should be run on the primary node:

```
# /sbin/cluster/utils/drbd_tool --action=resize --application apache
```

how to check? /cat/drbd?

Once the resize has been successful, the devices should be stopped:

```
# /sbin/cluster/utils/drbd_tool --action=down--application apache
# ssh server2 /sbin/cluster/utils/drbd_tool --action=down \
  --application apache
```

18.3.3 Off-line File system Reduction

Currently reduction in file system sizes can only be done as an off-line process unfortunately. This is because of two reasons:

- None of the Open Source file systems available for Linux actually support on-line reduction in file system size.
- The DRBD device only supports the increase in the size of a volume, not decrease.

Luckily actually reducing the size of a file system is quite uncommon!

In this example the logical volume “/dev/apachevg/lv01” is to be reduced from 256MB down to 128MB. The first step is to ensure that the application is not running:

```
# clstat -A apache
Cluster: sl4cluster - UP

Application      Node      State  Runnig  Monitor  Stale  Fail-over?
  apache         N/A      DOWN   N/A     N/A     N/A     Yes
```

Since the application is not running - even though the cluster is up and running - the file system is temporarily mounted and backed up:

```
# mount /dev/apachevg/lv01 /tmpmnt
# cd /tmpmnt && tar cpzf /tmp/apache_backup.tgz .
```

The file system is then un-mounted and the logical volume resized to 128MB:

```
# cd / && umount /tmpmnt
# lvreduce -L 128 /dev/apachevg/lv01
WARNING: Reducing active logical volume to 128.00 MB
THIS MAY DESTROY YOUR DATA (filesystem etc.)
Do you really want to reduce lv01? [y/n]: y
Reducing logical volume lv01 to 128.00 MB
Logical volume lv01 successfully resized
```

At this point the file system must be re-created and the contents restored:

```
# mkfs -t xfs -f /dev/apachevg/lv01
meta-data=/dev/apachevg/lv01      isize=256      agcount=8, agsize=4096 blks
=                                   sectsz=512
data      =                       bsize=4096    blocks=32768, imaxpct=25
=                                   sunit=0       swidth=0 blks, unwritten=1
naming    =version 2              bsize=4096
log       =internal log          bsize=4096    blocks=1200, version=1
=                                   sectsz=512    sunit=0 blks
realtime  =none                   extsz=65536   blocks=0, rtextents=0
```

The “-f” option must be placed after the “-t xfs” option in the above command. The “-f” (force) option is used since the existing file system still exists on the logical volume.

```
# mount /dev/apachevg/lv01 /tmpmnt
# cd /tmpmnt && tar xzpf /tmp/apache_backup.tgz
# cd / && umount /tmpmnt
```

At this point the file system on “ServerA” has been modified, but that on “ServerB” has not - so reduce the size of the logical volume on that server:

```
# ssh sl4s2 lvreduce -L 128 -f /dev/apachevg/lv01
WARNING: Reducing active logical volume to 128.00 MB
THIS MAY DESTROY YOUR DATA (filesystem etc.)
Reducing logical volume lv01 to 128.00 MB
Logical volume lv01 successfully resized
```

At this point the contents of the file systems for the application must be invalidated. Since only “/dev/apache/lv01” has been changed, that file system can be targeted directly. Firstly the DRBD devices for the application should be started:

```
# /sbin/cluster/utils/drbd_tool --application apache --action=start
# ssh sl4s2 ssh sl4s2/sbin/cluster/utils/drbd_tool --application apache \
--action=start --noprimary
```

Now force the remote copy to be invalid:

```
# /sbin/cluster/utils/drbd_tool --application apache --action=invalidate_remote \
--vg apachevg --lv lv01
```

At this point examining the local “/proc/drbd” device should show synchronisation taking place:

```
# head /proc/drbd version: 0.7.10 (api:77/proto:74)
SVN Revision: 1743 build by root@sl4s1, 2005-06-07 05:38:31
0: cs:SyncSource st:Primary/Secondary ld:Consistent
ns:4412 nr:0 dw:0 dr:4412 al:0 bm:8 lo:0 pe:26 ua:0 ap:0
[=>.....] sync'ed: 6.3% (126764/131068)K
finish: 0:00:57 speed: 2,152 (2,152) K/sec
```

Keep checking this file and once the synchronisation is complete, stop the DRBD devices:

```
# ssh sl4s2 /sbin/cluster/utils/drbd_tool --application apache --action=down
# /sbin/cluster/utils/drbd_tool --application apache --action=down
```

Now the application must be re-built using the following commands:

```
# clbuildapp -A apache -V -F --vgbuild
```

Since the application configuration has not been changed the “--build” option is not actually necessary - though it must be performed to ensure the file system size change is noted by the cluster software.

```
# clbuildapp -A apache -V -F --build
```

At this point start the application and check file system size:

```
# clstartapp -A apache
INFO 23/06/2005 07:02:17 Applications start completed successfully
```

```
# df
Filesystem          1K-blocks      Used Available Use% Mounted on
/dev/hda1            2016016    1798280    115324  94% /
none                 128020         0     128020   0% /dev/shm
/dev/drbd0           126272         316     125956   1% /apache
```

File system re-sized as appropriate.

18.4 Modification of application parameters

The administrator can modify the configuration file for the application at any point, though currently any changed values will not be taken account of. For example, if the administrator changed the IP addresses or network defined for the application only when the application is restarted will the changes take affect.

However the administrator should note that as soon as the configuration files have been updated they should run the “clbuildapp” script with “--vgbuild” and “--build” options. Failure to do so will result in invalid checksums being kept meaning many commands will fail to run.

19 Easier Application Management

The purpose of this section is to explain some additional application configuration directives that can be used to aid cluster management, particularly if many applications are configured.

Early in the development of Linuxha.net additional optional directives were added to enable the cluster to attempt to manage multiple applications in a logical manner. Although significant improvements to these directives may occur in future releases they provide a frame work covering auto-start of applications, preferred application nodes and even inter-dependencies between running applications.

Currently the following example configuration file for an application shows the three new optional directives in use. Each is completely independent of the other and is entirely optional. The new directives are highlighted in *italics>*.

```
<global>
  <version>0.2</version>
  <name>mysql</name>
  <takeover>force</takeover>
  <dependencies>samba</dependencies>
  <preferred_node>LEAST_CPU_LOAD</preferred_node>
  <autostart>true</autostart>
</global>
```

The table below summarises each of the new directives:

Element	Purpose
Global.dependencies	This should be a comma separated list of other applications that need to be running before this application is started.
Global.preferred_node	This is either a node name in the cluster, or either of the values "LEAST_APP_LOAD" or "LEAST_CPU_LOAD". It defines the host where the application should be started, (if both nodes are available).
Global.autostart	This should be set to "true" or "false" (though "yes" and "no" are also accepted). It indicates whether or not the application should be automatically started when the cluster is formed using the "clform" or "clstart" commands.

It is important to remember that these directives are only taken notice of by the "high level" cluster tools – that is the "clform", "clrunapp" and "clstart". Using the "clstartapp" will ignore these settings and simply start the specified application on the local node.

19.1 Differences During Application Build

If these additional directives are present during the build of an application, then some additional output is included. If the “autostart” parameter is included then a line similar to the following will be shown when using verbose output:

```
INFO 27/03/2004 10:30:06 Autostart parameter present and valid.
```

Similarly if use of the “preferred_node” is made then a line similar to following will be shown:

```
INFO 27/03/2004 10:30:06 Preferred_node parameter present and valid.
```

Finally if a “dependencies” attribute is included for each dependent application a line similar to the following will be shown:

```
INFO 27/03/2004 10:30:06 Validated dependent application samba.
```

Because these settings do not stop an application from being started via the “clstartapp” command if there are any problems it will only cause warnings to be issued rather than errors.

If the administrator does define dependencies then a check is made to ensure the builds of each of those applications is valid, and if not a warning is issued. Again this does not abort the build process.

If a dependent application can not be validated when the package is started using “clrunapp” then it will not be started, which might cause unexpected results.

19.2 Using the “clrunapp” Utility

Much effort previously in this document has been given over to describing the use of “clstartapp” to start applications – though in reality it is more likely that applications will be started via the “clrunapp” command or even automatically when the cluster forms using the “clform” or “clstart” utilities.

If using the attributes “preferred_node” and/or the “dependencies” attributes for an application then ideally this application should be started via “clrunapp” to ensure this additional information is taken advantage of and the application is formed in the expected manner.

Consider the application mentioned at the start of this section:

```
<global>
  <version>0.2</version>
  <name>mysql</name>
  <takeover>force</takeover>
  <dependencies>samba</dependencies>
  <preferred_node>LEAST_APP_LOAD</preferred_node>
  <autostart>>true</autostart>
</global>
```

This configuration indicates that the application “mysql” has a dependency on the “samba” application. It also indicates that the node that “mysql” is run on is the node with the least number of active packages when it is started.

If “samba” and “mysql” were the only two applications in the cluster, and the cluster consisted of nodes “serverc” and “serverd”, then consider the consequences of running the following command:

```
# clrunapp -application mysql
```

If this command was run on “serverc” then the output it would generate would be similar to the following:

```
INFO 27/03/2004 11:17:06 Validated cluster configuration.
INFO 27/03/2004 11:17:06 Successfully connected to cluster cluster2.
```

```

INFO 27/03/2004 11:17:06 Verified that application mysql is registered.
INFO 27/03/2004 11:17:07 Current application state : DOWN
INFO 27/03/2004 11:17:07 Application mysql depends on: samba
INFO 27/03/2004 11:17:07 Application samba will be started on node serverc
INFO 27/03/2004 11:17:07 Starting samba using command:
INFO 27/03/2004 11:17:07 /sbin/cluster/clstartapp --application samba --maxdelay 30
--verbose
INFO 27/03/2004 11:17:20 Application samba started after 13 seconds.
INFO 27/03/2004 11:17:20 Application mysql will be started on node serverd
INFO 27/03/2004 11:17:20 Starting mysql using command:
INFO 27/03/2004 11:17:20 /sbin/cluster/clstartapp --application mysql --maxdelay 30
--verbose
INFO 27/03/2004 11:17:34 Application mysql started after 14 seconds.

```

Since this is a high level command it will always produce verbose output – though only to the screen – it does **not** have a log file like the underlying utilities. Notice that the “samba” application has been started on the local node, whilst the “mysql” application has been started on “serverd” - since when it started the “LEAST_APP_LOAD” attribute meant that the “serverd” was instead chosen.

Please note that the “clrunapp” supports a limited range of command line options to help take account of these additional attributes, including:

Option	Purpose
--nodeps	When starting the application do not attempt to start any applications that are registered as dependents of this application.
--nologging	Indicates that the “--verbose” flag should not be passed to the underlying call to “clstartapp”. Usually it is meaning that the application start-up and “Lems” daemon log messages – which is most cases is the required action.
--localonly	Currently this is not used by the underlying “clstartapp” utility but in future releases, (probably 0.7.0 onwards) it will indicate that the application should not attempt to contact the other node allowing for faster application start-ups.
--node Y	Indicates that the application should be started on node “Y” - assuming that the node is available. This overrides the “preferred_node” setting if the specified application has one.
--preview	Do not actually start any applications – simply give a summary of what steps would be taken, (particularly useful when an cluster has lots of applications and dependencies).
--timeout	The maximum amount of time to wait for an application to start – overriding the value given in each application configuration file, (if present). This defaults to 30 seconds if not included on the command line or in the application configuration.
--reset	Reset the list of valid nodes for the named application, (allowing it to start on either of the two cluster nodes).
--nolocking	Indicate that the application should not attempt to communicate with the lock daemon when assigning shared resources.
--nochecksums	Ignore application and cluster configuration file checksum errors - only use in emergencies!
--force	Force the application to start by passing the “--force” option to the “clstartapp” command. It will ensure that the application starts in more circumstances than normal.
--reallyforce	Really force the application to start – even if the node it is running on does not have a valid copy of data (dangerous). [Currently not supported - may be removed for 1.0.0]

19.3 Using the “clform” Utility

As shown earlier in the document, (see page 74), the “clform” utility can be used instead of calling “cldaemon” on each node to form the cluster. The full list of supported options are shown below. Previously just the cluster start-up was shown. However, if the “autostart” attribute is present for any application then this utility will also automatically start the applications by calling

the “clrunapp” utility. Hence formation of the same cluster above would have resulted in output similar to the following:

```
# clform
output from live environment
```

In the above example simply running a single command resulted in the cluster forming and two applications starting – all within 45 seconds.

Version 1.0.0 of Linuxha.net provides the following command line options for “clform”:

Option	Purpose
--noapps	When forming the cluster do not attempt to start any applications that have been registered to autostart.
--force	Force the formation of the cluster. This will mean that the cluster will form even if a node is not available, and even if the nodes have a time difference larger than 10 minutes. In this instance the “--force” option is also propagated to any applications that are started via the “autostart” attribute.
--config X	Specify the name of an alternative configuration file – typically not used, though useful for debugging and testing.
--join	Rather than both nodes attempting to join the cluster a node currently not in the cluster should attempt to re-join it (see the example below).
--timeout	Override the configured cluster form time out and use this number of seconds instead.
--nolocking	Indicate that the cluster daemons and applications should not attempt to make use of cluster lock daemon.
--nochecksums	Ignore errors in cluster and application checksums - use only as a last resort.

As with “clrunapp”, the use of “--nochecksums” is strongly discouraged. Use of this feature should only be performed as a last resort since without the checksum the results of any of the Linuxha.net can not be guaranteed.

if the administrator attempts to form a cluster which is already running the utility will simply exit, thus having no impact on the state of the cluster or any currently running applications. In such cases the output from the command will be along the following lines:

```
INFO 30/03/2004 19:00:16 Validated checksum for cluster configuration
INFO 30/03/2004 19:00:16 Checking that the cluster is not already running...
ERROR 30/03/2004 19:00:16 Cluster cluster2 is already running.
```

19.3.1 Joining an Existing Cluster

One of the most useful options available for the “clform” command is the “--join” option. This allows a node to rejoin the cluster very easily. For example, if the “cldaemon” dies on a node, the following scenario is possible, as reported by “clstat”:

```
clstat output where node daemon has died
```

Notice that although the node “serverd” is marked as “UP” the “No daemon” message indicates that cluster services are not present on that node - even though it appears the “mysql” application is running there. This scenario is possible if the cluster daemon has aborted (due to error) or has been killed off.

The above scenario was considered as part of the cluster software design and so an application can actually run without issues if such a situation does occur. Of course it does mean that failover many fail-over conditions will not be dealt with automatically on this node until a daemon is restarted.

As mentioned previously (see page 71), it is possible to manually start the “cldaemon” process on each node to join the cluster, the command being similar to the following:

```
# cldaemon --join --verbose --detach
```

Of course this command must be run on the server which is currently not in the cluster. In most cases the “clform” command is preferred. Since it is classed as a “high-level” command it can be used as an alternative when a node is to re-join a cluster.

Using “clform” the command to use is simply:

```
# clform --join
```

When run on either node it will work out which node is in the cluster and which is not and run the appropriate command on the appropriate node and report back once the node has joined the cluster, or failed to join it:

```
show node joining the cluster
```

If “clform” is used when both nodes are down, or when both nodes are up a suitable error message will be shown, for example:

```
INFO 30/03/2004 19:00:19 Validated checksum for cluster configuration
INFO 30/03/2004 19:00:19 Checking cluster status...
INFO 30/03/2004 19:00:19 Both nodes are currently up - no action taken.
```


20 Performing Software Upgrades

20.1 Background Information

The aim of this section is to describe the recommended process that should be used to perform several common software upgrades that are likely to be necessary during the life time of the cluster.

As with the rest of this documentation the approaches taken aim to minimize or remove any downtime requirements for the clustered applications. The aim in all cases is to ensure client access remains completely unaffected unless absolutely possible. The situations covered here are;

- *Upgrading a clustered Application*
Although this doesn't really impinge on Linuxha.net administration, obviously there is an overlap - especially if multiple applications are currently part of the cluster configuration.
- *Upgrading Linuxha.net Software*
Although the software is now past at version 1.0.0 updates are expected to happen on a regular basis. The steps described here ensure unless otherwise stated for a particular release it should be possible to perform such updates with no loss of client access.
- *Updating Operating System Software*
Depending on the software components that are being upgraded it may or may not be possible to perform such updates online. This takes advantage of Linux's ability to replace binaries that are currently in use for many upgrades.

20.2 Upgrading Clustered Applications

Earlier in this document, and as covered in more depth in the Application Installation and Configuration Guides, it is possible to install Linuxha.net applications in the manner most appropriate for the application in question, and the administrators requirements. However typically applications will either be installed locally to each node, or installed as part of the replicated storage.

- *Upgrading Applications Installed on Local Storage*
If an application is installed local to each host it reduces the impact of performing upgrades - assuming that the data remains compatible between application versions.

In such cases the node where the node is **not** currently running on can be upgraded, whilst the application is running on the other node. Prior to the upgrade the administrator is recommended to make use of the "cldaemonctl" command to ensure that the application will not fail-over to this node, for example:

```
# cldaemonctl --msg "SETVALIDNODES APP=apache \  
NODES=servera FORWARD=yes"
```

The above indicates that if the application is running on "servera" a software failure will shut it down rather than attempting to start it the other cluster node (i.e. the one where the software upgrade is currently being performed on).

If this is the only application in the cluster, then the alternative is to remove the node being upgraded from the cluster:

```
# clhalt --node serverb
```

This ensures software and hardware failures do not fail over to "serverb" in case this disturbs the software upgrade that is taking place here.

Following a successful software upgrade on “serverb”, the node should be added back into the cluster, or the list of valid nodes for the application reset, depending on the method used, for example:

```
# cldaemonctl --msg "SETVALIDNODES APP=apache FORWARD=yes"
```

or:

```
# clform --join
```

Once the software upgrade has been completed on that server at some point the administrator should fail-over the application to the other node, and upgrade it in the same manner described above. An alternative approach (assuming the new software uses the same data format) is to wait until it actually fails to the other node and then update the software on the original one.

➤ *Upgrading Applications Installed on Replicated Storage*

Online update of software this is stored on the shared storage is probably not possible in most cases. In all cases the first step the administrator should perform is to ensure that any process monitor functionality is disabled. Usually pausing the Lems monitoring completely for the application is the easiest way to achieve this:

```
# lemsctl --application apache --msg PAUSE
OK
```

At this point the administrator is free to stop the application and perform the upgrade - obviously on the node where the replicated file systems are currently mounted. This approach has the advantage that the installation should only need to be done once, rather than once for each node. However the administrator should take care; many applications hard code some configuration files. For example Oracle may install into any location, but may expect an “/etc/oratab” file to exist.

Thus the administrator may need to check for files being installed or altered on local storage and manually copy such files to the remaining node in the cluster.

Following the update the application can be restarted and the Lems monitoring resumed:

```
# lemsctl --application apache --msg RESUME
OK
```

20.3 Upgrading Linuxha.net Software

In most cases it is actually possible to upgrade the Linuxha.net software without actually stopping the cluster! However as always the best approach the administrator can take is to always stop the cluster. The steps necessary for each approach are listed below.

➤ *Complete Off-line Software Upgrade*

Assuming that downtime has been agreed for all currently running applications the administrator must stop all clustered applications and cluster daemons. This is done using the command:

```
# clhalt --force
```

Once the cluster software has been stopped the administrator is now free to install the new software versions on each node.

Please take care to read the release notes prior to performing software upgrades since certain upgrades may require manual steps before the new software version is started.

If (and only if) the checksum algorithm has changed (it does so purposefully occasional), then the cluster and application configuration files will need to be rebuilt; for example;

```
# clbuild -V -F
# clbuildapp -A apache --vgbuild && clbuildapp - A apache --build
```

At this point the cluster can again be formed:

```
# clform
```

Of course following any upgrade the administrator is recommended to be careful attention to the following log files after cluster formation, and start-up of any applications;

```
/var/log/cluster/clnetd.log
/var/log/cluster/cldaemon.log
```

➤ *Partial Off-line Software Upgrade*

This approach is supported unless otherwise stated in the release notes for a particular version.

This approach attempts to upgrade the cluster software without taking the whole cluster down. Hence it is necessary to ensure that at some point all the applications are running on a single node.

For example assuming all applications are running on “serverb”, then take “servera” out of the cluster ready to perform the upgrade;

```
# clhalt --node servera
```

At this point the Linuxha.net software can be upgraded on “servera”. Afterwards this upgraded node can rejoin the cluster;

```
# clform --join
```

Now the applications that are currently on “serverb” should be migrated across to servera:

```
# clhalt --node serverb --action=failover
```

Now the software upgrade can be performed on “serverb”, and once complete this node can rejoin the cluster:

```
# clform --join
```

➤ *On-line Software Upgrade*

It is possible to upgrade the cluster without failing over any applications - as long as the release notes indicate the checksum calculation has not been changed.

This is the best approach for environments that need 24x7 live applications, since the applications can run on both nodes before, during and after the upgrade without interruption.

To do this choose either node and run the following command (where “clustername” is the name assigned to the current cluster):

```
# ps -ef | grep -- -clustername
root      482      1  0 Jun23 ?        00:00:00 cllockd-s14cluster
root      486      1  0 Jun23 ?        00:03:27 clnetd-s14cluster
root      497      1  0 Jun23 ?        00:02:04 cldaemon-s14cluster
```

This will show the process ID's for the lock daemon, network daemon and main cluster daemon. Use the “kill” command to kill off the cluster daemon, network daemon and lock daemon - in that order. For example:

```
# kill 497
# kill 486
# kill 497
```

Check the cluster using the “clstat” command at this point:

```
# clstat
Cluster: sl4cluster - UP

      Node      Status
      s14s1     DIED
      s14s2     UP

Application      Node      State  Started  Monitor  Stale  Fail-over?
  apache         s14s1  STARTED  1:01:52  Running    0      No
  samba          N/A    DOWN    N/A      N/A      N/A    No
```

So the application is still running on “sl4s1”, but the cluster daemons are not. At this point the software on server “sl4s1” can be upgraded, for example:

```
# cat /etc/cluster/VERSION
0.9.2
```

Now install the software:

```
# tpininstall -i -p linuxha
```

Now check the version again:

```
# cat /etc/cluster/VERSION
0.9.3
```

At this point the software should be restarted on “sl4s1”:

```
# clform --join
# clstat
Cluster: sl4cluster - UP

      Node      Status
      s14s1     UP
      s14s2     UP

Application      Node      State  Started  Monitor  Stale  Fail-over?
  apache         s14s1  STARTED  0:00:00  Running    0      Yes
  samba          N/A    DOWN    N/A      N/A      N/A    Yes
```

The same process can now be repeated on the second node. Following this the On-line upgrade will be complete.

20.4 Updating Operating System Software

The scope of the upgrade is reflected in the amount of interruption they may be necessary.

- *Non-kernel Upgrades*
For example updating of libraries or programs that are not dependencies of the Linuxha.net software have no impact.

However if updates are occurring to the following software care should be taken;

- Perl (or any standard Perl Modules)
- Core libraries (such as libc.so or libm.so)

In such cases the release notes for the updates should be checked for possible incompatibilities. If the changes are minor then performing the upgrade is very unlikely (though can not be 100% guaranteed) to cause no problems. Of course if in doubt any such upgrades should occur out of core operating hours.

- *Off-line Kernel Upgrades*
Such changes obviously require a machine reboot to take affect. Hence the safest way is

to;

Install software upgrades on both machines - steps necessary depend on distribution.
Shutdown the cluster - for example using "clhalt --force".
Reboot both servers - for example using "reboot".

Following the reboot the administrator is recommended to run the following commands on both nodes to ensure the "DRBD" kernel module is made available for this kernel version:

```
# cd /usr/src/cluster/drbd.7-10
# make clean
# make
# make install
```

At this point the administrator can start the cluster and any applications are necessary;

```
# clform
# clrunapp -A apache
```

Hence such upgrades should be done one server at a time. Prior to the upgrade of "servera" fail-over all applications

21 Handling Failure Scenarios

21.1 Introduction

The purpose of this section is to describe what happens when availability problems occur. The information contained here may help the administrator to decide which recovery actions are most appropriate to ensure application availability is returned as soon as possible to the user.

This section is highly dependent on the version of Linuxha.net that is in use. The section currently refers to version 0.9.0 and above. It is recommended that all sites run at least this version to ensure the information detailed here is accurate to their environments.

As well as describing typical scenarios the administrator may see, information on how the cluster handles such conditions is obviously included. Of course the aim here is that most conditions are found immediately and handled automatically by Linuxha.net. In many cases however there will be a short period of "downtime" - though how this affects the users depends not only on the problem that has occurred, but also on the application in use and obviously the speed of recovery, especially if recovery is not automatic.

It should be reiterated that as with most clustering software products, the design aim is to automatically handle and recover from failure of a single "component". Failure of multiple "components" should be noted, but in many cases may not be handled automatically.

21.2 Common Failure Scenarios

Although Linuxha.net only supports two node clusters presently (and thus may be considered "basic" by some groups), it does attempt to handle almost all single component failure scenarios automatically.

It is the responsibility of the administrator to check the cluster logs on a regular basis to capture information on potential problems (or problems that have occurred, but have been recovered from).

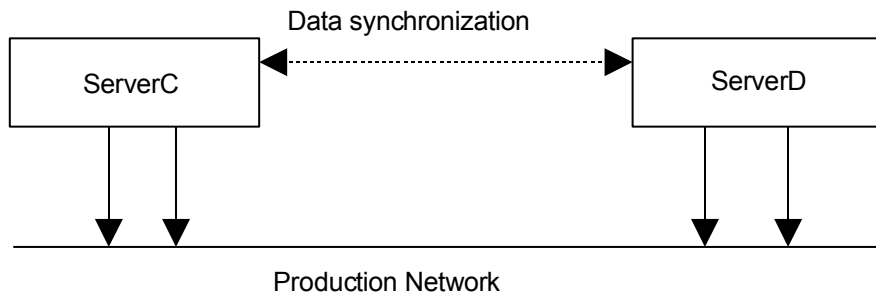
Version 1.0.0 does not provide hooks to alerting software. This functionality will be added in later releases.

The flexibility that keeping data replicated rather than shared does unfortunately mean that they may be more recovery scenarios to consider compared to classic clusters which the highly available data is instead shared.

From experience the following sections describe the most likely problems the administrator will encounter whilst running **any** cluster (though problems will of course discuss such failures in relation to Linuxha.net in particular).

21.2.1 Loss of a Network Link

The impact of this is really dependent on the resilience of the cluster network topology. If the cluster has spare cards in the network in question then it will automatically fail-over to an alternative card. In many cases this is completely transparent and clients will be unaware of this network change.



The actual steps, implementation details and short-comings of this type of failure and recovery are now considered in detail.

Detection of Link Failure

Under most conditions Linuxha.net makes use of the cluster network daemon to monitor all physical interfaces that are in use (i.e. have an IP address assigned currently), and take appropriate actions if a link failure is detected.

The method of detection depends on what each card physically supports - it will use calls support by "ethtool" or MII status checking. Any combination of cards supporting either method is fully supported.

However link failure checking is not possible for all card types - though this is currently thought to be confined to wireless Ethernet.

Actions on Link Failure

When a failure is detected then any IP addresses that are associated with the card in question will be migrated to the alternative interface - if available. The original interface is then marked as "faulty" for a short period of time.

If a further link failure occurs and the other alternative cards have been used, then the IP fail-over will only occur if the time period for any interface to be marked as "faulty" is passed. In that case it will reuse the interface, otherwise no interfaces for that network are deemed suitable.

When the cluster network daemon is unable to find an alternative interface for a failed network link then it informs the cluster daemon that this particular network has failed locally.

In such situations the cluster daemon will scan the list of currently hosted applications any any that make use of this network are automatically failed-over to the other node in the cluster - assuming it is operational. If a cluster is configured with several networks any application running not making use of the failed network is left alone.

Implementation Specifics

The cluster network daemon is only responsible for monitoring physical link problems - thus it will probably detect (and attempt recovery from) the following type of failures;

- Network Card failure
- Network Cable failure
- Network Port

It will not notice problems that occur at the IP or TCP/IP levels, such as;

- Host routing table issues
- Router failure
- Network reachability

If a fail-over is detected the recovery time is typically less than 250ms. All IP addresses associated with the failed card are migrated (physical and alias'ed addresses). For all IP addresses a gratuitous ARP packet is sent out to ensure associated network infrastructure can update their ARP tables with the network MAC address associated with the affected IP addresses.

Administrator Recovery Actions

Following a failure the administrator should obviously attempt to fix the problem. If this is a hardware issue on a node then the following actions would need to be followed;

- Migrate remaining applications off of the affected node
- Halt cluster services on affected node
- Halt node (if necessary) and perform necessary maintenance.
- Start the server (if necessary)
- Allow the node to re-join the cluster

The Halt and Re-start of the server hardware are only necessary if the server does not support hot-plug functionality for the component being replaced.

The commands necessary to do the above (assuming "servera" is to be brought down, and the commands are being run on "servera"):

```
# clhalt --node servera --action=failover
# halt
```

Perform necessary changes and Boot server;

```
# clform --join
```

If a server reboot is not necessary (for example just a cable needed to be replaced), then the administrator must ensure the network daemon re-scans all networks to ensure monitoring of the particular network in question is re-started. This can be done using the following command:

```
# clnetd --msg "REREAD"
```

After this monitoring of the failure network will again take place. Following this command check the network daemon's log file for any remaining problems.

The above command must be run on the node where the network problem occurred, since "clnetdctl" only communicates with the local cluster network daemon.

21.2.2 Handling IP-level failures

Since the network daemon only has scope for handling physical connectivity issues, the administrator can optionally ensure that a Lems module is used to scan connectivity on a per-application basis. The module type to handle this is "ip_module" - and a lems entry for any application will appear the same (apart from the application name which is shown in **bold**):

```
<check>
  <name>ip</name>
  <type>internal</type>
  <module>ip_module apache</module>
  <interval>10</interval>
  <action_list>
    <action rc="0" action="NOP"/>
    <action rc="1" action="RUN move_ip"/>
    <action rc="2" action="STOP"/>
  </action_list>
</check>
```

Notice that the IP checking performed here is determined by the contents of the application configuration for the application in question. For example if the "apache" application had the following information in its "appconf.xml" file:

```
<networks>
  <network net="main"
    ip="172.16.177.200" netmask="255.255.255.0"
    checklist="172.16.177.1" checkpercent="100"/>
</networks>
```

In this case the "checklist" and "checkpercent" environment is used every 10 seconds. If problems do exist then the module will communicate with the cluster network daemon indicating a network problem.

Currently the software typically only tries to test IP handling every 10 seconds, though this can be reduced down to 1 second if necessary.

The administrator should be aware that this module has the ability to affect every application - since the IP fail-over will migrate all IP addresses associated with a network card, not just those for a particular application.

21.2.3 Failure of Data Replication Network Connection

The network that is used for DRBD data replication is handled in the same way as any other network configured in the cluster - that is IP fail-over to an alternative card, if available, will be performed.

If a dedicated network is not assigned for such communication then client/server communication as well as the server-server DRBD traffic will be mixed on the same card. This will work without problems but is not ideal, especially in production systems.

The more usual scenario is to have a dedicated network for such traffic. If stand-by cards are not available, or even IP fail-over events have occurred for monitoring to cease on this network, then obviously Linuxha.net functionality will be affected.

In the case of connectivity loss for this network then if the data is currently synchronised then the applications will continue to function, though the remote data copy will become "Stale". In such cases following problem resolution the stale data will automatically be updated after a short period with no administrator intervention being necessary.

However if a previous error condition has resulted in the local data copy being stale then such a loss will lead to I/O errors being returned to the application. If this causes the application to fail then the process monitor may fail the application to the other node, though otherwise the administrator must take steps to intervene.

21.2.4 Application Software Failure

For a cluster to function effectively the administrator must ensure that the application software is monitored as well. The most common way of doing this is to simply ensure that certain processes are running, and if not take some remedial action – such as attempting to restart the application, or even failing over the application to the other node in the cluster.

Since process checking is very common there is already a module available for the Lems sub-system that can be used to monitor processes – known as the “procmon” monitor (examples of which have been shown previously). Detailed information on the configuration and use of the “procmon” module can be found starting on page 184.

Of course simply monitoring for certain processes is not the only way to validate the health of the application. Other alternatives (or additions) include:

- Test transactions – running a fake request against the application service to see if it responds as expected.
- Log checking – checking the tail of a application log file looking for serious errors that may indicate an application problem.

The Lems sub-system can be easily extended to include such tests – see the technical details on the sample transaction-based monitor for Lems starting on page 188 if the administrator wishes to investigate such possibilities.

However the rest of this section will concentrate on how the “procmon” monitor is used and how it impacts application availability. First consider the following extract from the status of a sample “samba” application:

```
# clstat --application samba
...
Process Monitors

      Name      Status  Restarts   Current      Reset at
      smbd      Running    3         0            N/A
...
```

A process monitor works on the principal that it is better to attempt to restart an application on the local machine if it fails rather than initially attempting a fail-over – the reason being that a fail-over of the application to the other node increases the period of the application being unavailable.

In the above example the application is able to be restarted up to 3 times currently. If the “smbd” process is killed, (the one which is being monitored in this example) then shortly afterwards it will be restarted and running “clstat” again will show something similar to the following:

```
Process Monitors

      Name      Status  Restarts   Current      Reset at
      smbd      Running    3         1  23/02/2004-16:55
```

This indicates that the application has been restarted once out of a maximum number of three times. The “reset at” column indicates when the count of restarts is set back to 0 – it is defined as an interval after the last restart – and is typically set for a couple of hours.

In the above example the Lems log file will include an indication of the restart has occurred:

```
INFO 23/02/2004 15:55:45 Restarting the application (/samba/cfg/scripts/restart) ...
Error: Unable to find process to kill
```

In the above example the “restart” attempts to kill a running process first – which of course was not in place – hence the error message.

21.2.5 Stopping Fail-over (from application monitoring)

When initially building a cluster, or are changing it at a later date, it often makes sense to disable this application level monitoring that Lems offers. There are three ways in which this can be done:

- Change the Lems configuration file for the application to ensure the application monitor is not included - and then kill off the Lems daemon for the application in question.
- Create an ignore flag on both nodes in the cluster.
- Send a message to the “Lems” daemon to stop scheduling the specified component.

The first approach will work because the main cluster daemon will restart the Lems daemon for any running application when it notices that it has died.

The second one requires that the “flag check” module is configured and in use - and if such is the case does allocate non-root users to control application checking. This can be advantageous in some circumstances.

Typically the flag check monitor uses the following directory for storing flags:

```
/etc/cluster/<applications>/flags
```

Thus to stop the “smbd” monitor the following command would be used:

```
# touch /etc/cluster/samba/flags/smbd
```

The name of the file is the name of the monitor you wish to stop – if the flag directory does not exist, simply create it. Once the flag has been created, the “clstat” command will show something similar to the following:

```
Process Monitors
```

Name	Status	Restarts	Current	Reset at
smbd	Stopped	3	1	23/02/2004-16:55

At this point the monitor has been disabled. It can be left disabled as long as necessary, though of course this is only recommended for “maintenance” periods on production-like environments.

When the administrators wishes to resume the monitoring simply remove the flag file:

```
# rm /etc/cluster/samba/flags/smbd
```

The alternative approach is to send a message to Lems. This can only be done as “root”, and in this case the administrator would use:

```
# lemsctl --application samba --msg "PAUSE smbd"
<check above command>
```

Later resuming the specific monitor requires the command:

```
# lemsctl --application samba --msg "RESUME smbd"
<check above command>
```

If an application monitor notices that the application it monitors has failed more than the maximum number of times, then the typical response is to fail-over the application to the other node – though this really depends on the Lems configuration for the monitor being something like:

```
<check>
  <name>smbd</name>
  <type>internal</type>
  <module>procmon /etc/cluster/samba/smbd.xml</module>
  <interval>10</interval>
```

```
<action_list>
  <action rc="0" action="NOP"/>
  <action rc="1" action="STOP"/>
  <action rc="2" action="FAILOVER"/>
</action_list>
```

If the action for return code "2" is not "FAILOVER" then the application will obviously not fail-over.

21.2.6 Process Monitor Administration

Current versions of the Lems monitoring daemon allow fine-grain control of the environment whilst the monitor for an application is actually running. For example it is possible to change the maximum number of restarts and the current number of restarts by making use of the "lemsctl" utility.

For example if the process monitor above is configured to use a maximum of 3 restarts then the following command could be run to increase this to 5 - **for this invocation only**:

```
# lemsctl --application samba --msg "CMD smbd MAXCOUNT 5"
OK
```

Although the count of current re-starts is reset after a period of time, using "lemsctl" it is possible to reset this count immediately:

```
# lemsctl --application samba --msg "RESET"
OK
```

For more information on controlling process monitors see the text starting on page 174.

One major limitation of earlier versions of "Lems" that has been addressed is that of application status management in the cluster configuration. If an application fails on a particular node and a fail-over is initiated by the node itself then a software flag is set on that node indicating that a fail-back to this node is not applicable for the application.

The purpose of this is to ensure that if a problem with a process monitor occurs and it fails over the application incorrectly, after failing between the two nodes it will halt the application. If this functionality was not available then the application would continue to fail-back between the nodes indefinitely which could not be considered as manageable.

This scenario typically only occurs happens when there is a software fault – either with the application itself or the configuration of one or more monitor processes for the application. However it is possible to also happen when there are consistent networking issues within the infrastructure that makes it impossible for a running application to find a suitable reliable interface on either node.

21.2.7 Node failure (Hardware or Operating System)

This is the scenario where one of the nodes in the cluster appears to "die" - that is it either shuts down at an unscheduled time due to unforeseen problems, or simply "dies" - i.e. suffers from a loss of power. The cause of the problem could also be software – for example a kernel panic still causes the server to be unavailable in the same manner as a processor or power supply failing¹⁰.

Such cases are now becoming more rare as Operating Systems become more stable, but still hardware problems can occur at any time, and of course this is the typical "disaster recovery" scenario when one of the server locations is lost due to fire, for example.

The next section covers handling of "Node Failure Scenarios" in more detail, but it is important to realise this is one of the most important abilities that all clustering software must deal with – including Linuxha.net.

¹⁰Though of course the recovery actions required are far different. Typically software failures may result in only short periods of downtime meaning that recovery of the cluster availability is usually sooner.

21.3 Managing Node Failure Scenarios

Here it is assumed that the cluster is running with two nodes – if the cluster is running with a single node and that node fails then obviously all application available will be lost until the administrator manually start either or both nodes in the cluster.

If the node that fails is not running any applications currently then **no loss of application services will occur**. In this case the loss of the node will mean that the remote copies of data will be marked as “Stale” meaning that a synchronisation of the data will need to occur once the failed node rejoins the cluster.

If the node is actually running some applications then these applications **will suffer a period of outage** whilst the other node in the cluster notices the node has failed and then considers the viability of starting each application on the remaining node.

The time taken for an application to be restated depends on the time-outs set for cluster communication and how many applications need to be started on the remaining node. Currently each application is started in sequence since this is deemed the most sensible approach when recovering multiple applications.

The application will only be started on the remaining node if it is deemed viable for that application. If the copy of the data on the remaining server is state then the application will only start if running with a “forced” start-up.

If a forced start of an application does occur, using potentially “stale” data, then when the other server attempts to rejoin the cluster its copy of the data **will be over-written – even though it was (previously) thought to be the (only) valid copy of data!**

Hence the importance of ensuring the administrator chooses the most suitable value for the “takeover” attribute for each package.

One further point the administrator should consider is that during the actual take-over the “clstat” command may indicate the application as “DOWN” for a short time period prior to the take-over completing.

21.3.1 Checking Application Status

Once the applications have been started on the remaining node in the cluster, (assuming each is viable), using “clstat” on the application would show a status similar to the following:

```
clstat output after hardware failure and application running
```

If at this point the cluster daemon logs on “serverB” (“serverA” being down) were checked, the administrator should see something similar to the following indicating that the service has failed over to “serverB”:

```
cldaemon log output for application fail-over
```

Notice that the “samba” application has 3 invalid logical volumes – viewing the detailed information for that application shows what the administrator should expect:

```
clstat output for application with stale data
```

21.3.2 Recovering Application Data Availability

When the above scenario occurs obviously high availability of the “samba” applicatino can not occur until the remaining node is brought back to into synchronisation – there is currently just a single copy of data, on a single node – so it is imperative to get the other node back into the cluster as soon as possible.

Once the other node has had any hardware and/or software problems resolved the administrator should boot it as normal. Because the cluster is already running it will **not** rejoin the cluster until you manually ask it to do so¹¹. Assuming this node is called “serverA” in this example, then the following command could be used to join the cluster:

```
# clform --join
```

This would have shown something like the following:

```
clform --join output
```

Once both nodes are back in the cluster, the output will appear similar as follows:

```
clstat output
```

¹¹ This is of course assuming that the “clstart” utility has not been configured to automatically start/join clusters on machine boot.

At this point if you check the "lems" file you will see the following entries for around the time the backup node failed:

```
fsmonitor output up to daemon contacted
```

Shortly after the node rejoins the cluster the data synchronisation should start. Again the Lems log for the application will contain some information to indicate the actions taken:

```
lems output showing synchronisation attempted
```

Following this output "clstat" can be used to view synchronisation progress.

```
# clstat --application samba  
verbose output showing sync in progress
```

It should be remembered that DRBD only synchronises regions of each file system that have changed and so quite often this completes before the administrator can even check!

If synchronisation does take some time to complete the Lems log may contain entries similar to the following:

```
log entries showing sync progress?
```

DRBD will synchronise all devices concurrently - thus the amount of bandwidth consumed on the network is not only determined by the synchronisation rate configured for the application, but also the number of file systems the application has (and how many contain stale data).

Once all data volumes for this application have been synchronised the important lines present in the "lems" log file are the following:

```
messages from lems log showing sync complete...
```

At this point the data is fully available again on both nodes.

It was noted above that the bandwidth to consume is a product of the number of file systems an application has. The actual time taken to synchronise all file systems thus depends on a number of factors, including;

1. The amount of stale data to synchronise – the longer an application runs on a single node, the larger the amount of information that is likely to be out-of-date on the failed node.
2. The number of file systems to synchronise – each file system is synchronised separately (though concurrently).
3. The performance of the host systems – the raw power of both systems is important during the operation of the cluster since both nodes are involved in handling both the data being synchronised and any existing file systems that are fully synchronised. This particular pertains to disk I/O - especially if multiple file systems are being synchronised.
4. The bandwidth of the network connection – obviously a 1Gbit/second connection offers a better throughput compared to a 100Mbit/second connection.
5. The current load on both servers – the synchronisation effort attempts to ensure that other I/O traffic on both servers is not unduly impacted – hence if the servers are busy doing "real" work the synchronisation rate is likely to be lower than the possible maximum.

All of these factors combine to produce a rate of synchronisation that does not remain static – hence the estimated completion times is purely that – an estimate – though it typically becomes more accurate once it has been running for at least 1 minute, (assuming other steady-state conditions).

21.4 Loss of Server Main IP Interface

Earlier versions of Linuxha.net relied on the "main IP" interface to perform many actions. For example, the high level cluster form routines and application build utilities used it solely, and if not available simply would not function.

Version 1.0.0 does not suffer from this limitation. Since it is strongly recommended to use at least two networks, (one for DRBD traffic and the other for client access), loss of the main IP connection does not cause major problems - though it may cause applications to fail-over, as described on page 150 earlier.

When multiple networks do exist the application components that usually make use of the "main IP" addresses (that associated with the hostname), these will simply attempt to use an alternate network address and continue to process as normal.

This alternate network probing relies on SSH-type functionality. Hence the administrator must ensure that all IP addresses are "known hosts" to SSH otherwise it may attempt to prompt - which may cause the program to appear to "lock".

Of course if **all** network links are lost between the two nodes then "network partitioning" occurs, which means that drastic action has to be taken - see the details on page 161.

The functionality of various commands when the main network address that is most often used is not available is shown below for completeness.

- *Cluster daemon* – if the cluster is already running then it may take a few moments for the cluster daemons to re-establish communication using another IP address. If the cluster is being formed the software automatically chooses an alternative IP address that is available and the cluster formation works as expected.
- *Application Start-up* – If the application is not running, then if you attempt to start it when

the main IP address for the remote node is not available, you are likely to see something like the following in the start-up:

```
Message showing alternative connection to other node during clstartapp
```

Notice that the final line indicates that another interface has been successfully found with a suitable level of communication for the start-up of the application to continue.

Of course if no interfaces exist that are suitable for hosting the application IP addresses on this host it will fail to start correctly.

- *File System Synchronisation* – If the primary address is also used for “DRBD” communication, then if the application is already running then the remote copies of the data will become stale. If the application is just starting the “Lems” log file will also contain entries similar to the following indicating the data is no longer synchronised:

```
lems messages if DRBD interface not available
```

Of course once communication to the main IP address is restored then any stale data will be quickly synchronised by a running “fsmonitor” monitor for the application in question:

```
messages showing synchronisation starting/completing
```

Of course all of the above scenarios are only possible if multiple IP networks are configured as part of the cluster topology. If that is not the case then the loss of the interface might result in “network-partitioning” - complete lack of communication being possible between both nodes.

21.5 Loss of Cluster Daemon

In rare cases it is possible that the Cluster daemon on a node may die, whilst applications may or may not be running on the affected node. It is important to realise that one of the design features of the software was to be able to gracefully handle this scenario with as little immediate impact to running application as possible. Indeed this feature is made use of when considering Linuxha.net software upgrades as described on page 145.

Typically if a software problem causes a cluster daemon to fail it will only affect a single node. On the node that is remaining the cluster log will show something like the following:

```
message when remote server has died
```

Hence the daemon understands that the daemon has died since the remote host is still contactable. In this case any applications running on either node are left undisturbed since the most likely cause of the problem is software on the node that it appears to have failed on.

The most extreme case would be when both cluster daemons are down, and indeed running “clstat” gives the following:

```
# clstat
Cluster: simon1 - DOWN
```

However, in this instance running “df” will show the clustered file systems as still being available:

```
# df -h output
```

Also the other cluster daemons may continue to run:

```
ps -ef | egrep "clnetd|cllockd" output
```

In such cases when of the standard commands unfortunately will not function - for example consider “lemsctl”:

```
# lemsctl --application apache --msg "fred"
output when both cluster daemons have died
```

To recover from this scenario simply “form” the cluster again:

```
# clform --force
text showing cluster forming
# clstat
clstat output
```

Notice that the cluster daemon has determined that an application appears to be already running, and has updated the application status information it keeps to reflect this. The cluster daemon log file should reflect this as well:

```
message showing found application for cluster log
```

Even if the cluster did not recognise the application was running, it would notice the IP address was in use and thus not actually attempt to start the application itself, even if explicitly configured to via an auto-start option.

For each application that is considered to be running the cluster daemon will probe for a corresponding running Lems daemon, and if not found will automatically re-start it.

21.6 Understanding Network Partitioning

Network partitioning is fortunately a rare condition - especially when multiple networks are defined in the cluster topology. For such a condition to occur all networks defined in that topology must cease to function.

In this situation each node in the cluster must attempt to understand whether;

- *Whether the remote daemon died?*
Could the failure to communicate with the remote server simply be down to the fact that the server has died? The cluster daemon will attempt to communicate with the other daemon using each network in turn until all fail or communication via an alternative route is found.
- *Are any application IP addresses available?*
For each defined application in the cluster that is considered to be running, if running on the other node then check to see if any IP addresses can be ICMP pinged for that application. If so the remote daemon has died, but the node is unaffected.
- *Does a running application define an IP checklist?*
If so then use it to ascertain whether network connectivity is available. If so then assume the remote node has crashed and attempt to swap applications.
- *Applications are no help - what next?*
If any applications are running on the remote node we must assume the node is down, otherwise assume partitioned.

When the remote node is down then the local node will start any applications that were thought to be running on that node. When the environment is set as “partitioned” then **no application state changes occur!**

Since the IP check-list details for each running application are used, it is strongly recommended that sensible entries be defined for each application, (using known good IP addresses, such as routers, switches and other hosts).

The worst case scenario is when applications are running but none have usable check lists, or the check lists are invalid. In this case the daemons may assume they each node are down, rather than the network being partitioned. This means that each node will attempt to start the other nodes applications!!

This action is not the correct thing to do but since the daemon does not have enough information to ascertain any other status it must assume the node is down to attempt to make applications that were started to continue to run.

Once the networking condition that causes the partitioning is resolved the daemons will re-establish communication automatically. They will notice that both nodes are running the same applications and one of the nodes will ***immediately reset itself***. This occurs without disk synchronisation and is intentionally abrupt in an attempt to remove the "split-brain" scenario - both nodes attempting to run separate copies of the same application.

The actual node that is killed is based on taking the first running application and killing off the server which started that application first. The check for duplicate application status is carried out approximately every 10 seconds.

The above situation is the worst case scenario that can be avoided most of the time by use of redundant links and ensuring each application has a checklist.

21.7 Data Consistency Issues

Sometimes after a particular event it may be wise to ensure that a forced data synchronisation takes place. This might be after a series of failures including failures whilst other synchronisation events were taking place.

This is particularly likely when using “ENBD” for replication, but for peace-of-mind is also available for “DRBD” configurations.

The synchronisation command below should be run from the primary server – running from the secondary server might work, but has not been tested fully. The command to run is as follows:

```
# clbuildapp --application X --force --verbose --sync
```

The force option is required since without it it will fail because typically this command is usually run on applications that have not yet been fully defined in the cluster.

When using “DRBD” sometimes when using the above command it will immediately complete – indicating that the synchronisation has not been performed. In this instance make use of the “--forcesync” argument to force the complete synchronisation, for example;

```
# clbuildapp --application X --force --verbose -sync --forcesync
```

Part IV:
Technical/Developer Information

22 Implementation details of “clstartapp” & “clhaltapp”

This is probably one of the most complex and widely used utility that will be used as part of the cluster software. The “clhaltapp” is simply a link to the “clstartapp”, since much functionality is shared.

The purpose of “clstartapp” is to start the specified application on the local node, whilst the purpose of “clhaltapp” is to stop the specified application on the local node. If the specified application is already running or already stopped then no action is taken (though this is only possible if the cluster daemon is running on the node on which the action is taken).

22.1 Supported Command Line Arguments

The “clstartapp” currently supports the following command line arguments:

Argument	Purpose
--application X	Specifies the name of the application to start.
--file	Where to send the output to (when verbose output is given, overriding the default name and directory provided by the cluster configuration file).
--force	Perform a force start - this is required if the cluster is running on a single server, (or the other server is not available for data replication).
--reallyforce	Perform a really forced start – this is necessary if you wish to start the cluster using STALE or potentially stale data.
--maxdelay N	The maximum amount of time when waiting to un-mount a file system before forcefully killing of the processes, (and waiting for the application to stop).
--checks	Perform the maximum amount of checking – results in a slower startup, but is safer. Recommended, especially if you have problems but can not identify them.
--verbose	Verbose mode - log messages to stdout.

When running as “clhaltapp” the currently supported command line arguments are:

Argument	Purpose
--application X	Specifies the name of the application to stop.
--force	Go through the steps of stopping the application, even if it does not appear to be running.
--maxdelay N	Maximum amount of time to attempt to un-mount a file system before forcefully un-mounting it. This also defines the length of time to wait for the application stop script to complete before killing the application.
--verbose	Verbose mode - log messages to stdout.

22.2 Default Argument Settings

If the “--maxdelay” option is not specified it will use the “application -> maxstarttime” and “application -> maxstoptime” settings from the configuration file for the package, (for starting and stopping the application respectively).

If the “--maxdelay” is missing and the configuration file does not contain default values, then a default value of 30 seconds is hard coded into the program for both starting and stopping.

If the “--file” option is not specified then a file in the directory specified for the cluster logging in the main “clconf.xml” file will be used (which must exist if you've built the cluster successfully). The name of this default file will be as follows:

For stopping an application:

```
clhalt.<application_name>.log
```

For starting an application:

```
clstart.<application_name>.log
```

If the start and stop programs defined for the application generate any output to standard out or standard error they this will **not** appear in the log file specified, instead these scripts have there own output location. Currently the following file names are used:

For stopping an application:

```
/var/log/cluster/<application_name>.start.log
```

For starting an application:

```
/var/log/cluster/<application_name>.stop.log
```

22.3 Ascertaining Cluster Status

Whether the utility is run as “clrunapp” or “clhaltapp”, the first major purpose of this code is to validate the status of the cluster. This is done using the “remote_node_available” function – which returns a list with two items, the values of which are defined as follows: :

Return Value	Meaning
0, undef	The cluster is up and running - both nodes are available and the IP connectivity used for file system synchronisation is working as expected.
1, undef	The remote node appears to be completely down.
2, name IP	The remote node is partially up, (not supporting synchronisation channel). It can be communicated with using the returned name or IP address. (Communication is ssh protocol).
3, IP	The node can be pinged on the specified IP address, but no channels offer ssh capability.

This function is called with two arguments: the name and IP address used to replicate the data.

The algorithm used has been greatly simplified since the original inception of this project – this is due to practical states that are likely to occur in the real-world environments, rather than attempting to deal with every possibility – which was overly complex and hence actually prone to making incorrect decisions.

22.4 Starting cluster packages - Condition Decisions

The output from the previous utility is then used as a basis for further tests to ascertain the status of the cluster. The purpose of these decisions is to further indication whether the local and / or remote data sources are current and hence whether the application can start at all, based on the command line argument supplied.

Before doing anything we ascertain the validity of the local and remote data by looking for “STALE” files in the following directory:

```
/cluster/control/.status/<application>
```

We then set flags to indicate whether the local node (the node on which we are trying to start the application) thinks the local and remote copies are valid or not. If available the information on the remote node is retrieved and compared. We also look for the “TIME” file on each node - this contains the UNIX time when the application was last started on that node, the most recent is believed if there is a discrepancy between the nodes.

Decision 1: Voting on Data Currency

The first decision is only acted on if the status of the cluster is clean - return value "0,undef" from the "remote_node_available" feature. This decision does the following:

1. Can we get a list of the status files for this package from the remote node? If not change status to "3" and take no further action in this decision.
2. Get details of the STALE flags from the remote node – do they agree?
3. If so then we believe the local node and set "ldata_current" and "rdata_current" to suitable values and finish the decision process.
4. Now the flags do not agree and so we attempt to get the TIME values from each node, (the time the package was last started on the cluster. If both have copies then we believe the STALE settings from whichever node is the most recent. If only one node has a TIME file then we believe that node.
5. Unless the "--force" argument has been specified then we abort – without it we believe whatever the local node indicates.

Decision 2: No Remote Host Appears Available

If the cluster consistency check returned "1,undef" it means that the remote node is appears to be down completely. if this is the case then the steps below will be taken:

1. Unless the "--force" option is specified then we abort. This is because if the remote node is not available it is not possible to guarantee 100% that the local copy of the data is the most recent, (or the STALE flags locally are actually the most up to date).
2. If the local data appears to be current, we indicate the remote node is not and start with a warning.
3. If the local node has indicated that it believes its local copy is not valid, then if "--reallyforce" has not been specified an error is given. If "--reallyforce" has been specified then we must set the local copy to valid, and delete all the STALE flags for the local node.

It should be noted that this decision is a very important one. This ensures that if the user has specified "--reallyforce" the local data copy will be used - **even if we know it is not up-to-date**. The impact of this is that when the remote node is added to the cluster it's copy of the data will need to be over-written.

Decision 3: Starting a package with no Data Replication Channels

This decision is called into affect when the return code from the "remote_node_available" is "2,IP". This means that it is not possible to access or update the remote data source, but it is possible to run "ssh" commands using the specified "IP" address.

The biggest point to note here is that if the local copy is out of date then we must only accept it as the valid copy (which is our only choice here), if the "--reallyforce" flag has been specified.

The steps taken by this decision are as follows:

1. If neither node has a TIME setting then we believe the local STALE flags, if any. If the --force option has not been specified then we abort (some risk we might be incorrect in assuming the local settings).
2. If there node has a TIME setting then we believe whatever flags that node currently has.
3. If both nodes have settings we believe whichever has the the most recent TIME status.
4. From the above points we will have set "ldata_current" and "rdata_current" according to the particular node. Now if the ldata_current is 0 then we abort -- unless the "--reallyforce" flag has been specified.

The above series of checks are attempting to ensure that when the data synchronisation channel is not available we only continue if the local node is current, the remote node is not currently hosting the application.

The "--reallyforce" flag will allow a non-current start up, but again should be used with caution.

Decision 4: Can ping node, but not communicate

1. If the local host thinks the remote node is the live node, and a ping returns OK, then we must be at some risk of not using the most up to date information – hence if the “--force” or “--reallyforce” options have not been specified then we abort.
2. If the host thinks that the local copy is valid, then we use it, and indicate that the remote copy is not valid.
3. If the local copy is not valid then we abort – unless the “--reallyforce” option has been set, and in this case we issue a warning and start the the local stale copy.

22.4.1 Actions on a Clean Start-up

Now that the decisions used to validate the status of the cluster have been showed a “clean” start up consists of the following steps:

- The “mkmdcfg” utility is called to generate an up to date RAID configuration file containing all details of the local NBD and LVM devices used for this particular package.
- If the remote data is classed as current the following actions are then taken:
 - A call is made to the remote server to run the “nbd_svrstart” utility to check to see if any NBD or ENBD servers need starting.
 - If necessary a call is made to the remote server to run the “nbd_svrstart” utility to start up all NDB or ENBD servers on the remote system.
 - The “nbd_clientstart” utility is run locally in check mode to see if any NBD or ENBD clients need starting.
 - If necessary “nbd_clientstart” is used again to start any clients necessary.
- A call is made to the “getmdlist” utility to get the details of the RAID and NDB mapping information to configure the RAID infrastructure.
- For each RAID device a call to “mkraid” is used, with the flags “--really-force”, “--no-delay” and “--dangerous-no-resync” to create each specific device.
- If the remote data is not considered current then the “raidsetfaulty” utility is called against the NBD or ENBD device for each RAID device used for the application.
- If the local data is not considered current then the “raidsetfaulty” utility is called against the LVM device for each RAID device used for the application.
- Now that all the RAID devices have been started, (with the local or remote devices faulted if necessary), the next step is to mount the file systems, which consists of the following steps, against each device taken from the “fsmap” file:
 - Get the type of file system expected and run “fsck -a” against it to repair the file system if necessary.
 - If any “fsck” returns a 4 then a reboot is required before the application can be used - and hence an error is returned and the package start aborted.
 - The “mount” command is used to mount the file system on the required mount point.
- Now that the file systems are mounted the next step is to assign the IP address of the package against one of the interfaces registered.
- The next step is to start the application using the script specified in the application configuration file, (or issue a warning if this script does not exist).
- The flag files indicating the status of the application are written to the local and remote nodes.
- If the file “/etc/cluster/<application>/lems.local.xml” exists then a “Lems” session is started to monitor the package locally.
- Lems is then responsible to monitoring the status of the package on the local node - see more on “Lems” and its interaction with the Cluster daemon later.

22.4.2 Actions on a Clean Shutdown

here

22.4.3 Actions on a non-Clean Shutdown (no remote available)

The key here is that once the file systems have been unmounted the “unsync” flags for all file systems will have been created. In this instance it is important that the cluster does not attempt to do anything with the remote node, so suitable “STALE” flags will be created (in the format “VG.LV.remoteserver.STALE”) once the application is started:

```
# ./clhaltapp --application apache --verbose --maxdelay 10
```

The format of the output will be similar to the following:

```
INFO 21/09/2003 11:03:56 Global section configuration validation complete
INFO 21/09/2003 11:03:56 Checked that node names resolve to IP addresses
WARN 21/09/2003 11:03:59 No response from remote node using ssh
WARN 21/09/2003 11:03:59 Ensuring the cluster is not forming - will take
approximately 10 seconds
INFO 21/09/2003 11:04:11 Checking heartbeats for any sign of life...
WARN 21/09/2003 11:04:11 Checking heartbeat serverb via ssh...
WARN 21/09/2003 11:04:14 Checking heartbeat serverb2 via ssh...
WARN 21/09/2003 11:04:17 Attempting UDP ping of 172.16.177.101...
WARN 21/09/2003 11:04:19 Attempting UDP ping of 172.16.177.111...
WARN 21/09/2003 11:04:21 Remote node is down
WARN 21/09/2003 11:04:21 Specified application stop script does not exist
INFO 21/09/2003 11:04:21 File system /apache/admin already un-mounted
INFO 21/09/2003 11:04:21 File system /apache/logs already un-mounted
INFO 21/09/2003 11:04:21 File system /apache/docs already un-mounted
INFO 21/09/2003 11:04:21 IP address 172.16.235.200 removed after 1 attempts.
INFO 21/09/2003 11:04:21 Attempting stop of RAID device /dev/md0...
INFO 21/09/2003 11:04:21 RAID device /dev/md0 stopped.
INFO 21/09/2003 11:04:22 Attempting stop of RAID device /dev/md10...
INFO 21/09/2003 11:04:22 RAID device /dev/md10 stopped.
INFO 21/09/2003 11:04:22 Attempting stop of RAID device /dev/md1...
INFO 21/09/2003 11:04:22 RAID device /dev/md1 stopped.
INFO 21/09/2003 11:04:22 Stopping all active NBD clients...
INFO 21/09/2003 11:04:23 All active NBD clients stopped.
WARN 21/09/2003 11:04:23 Remote server not responding - can not shutdown NBD servers.
INFO 21/09/2003 11:04:23 Removing LIVENODE control files...
WARN 21/09/2003 11:04:23 Remote server not responding - can not remove LIVENODE
files.
INFO 21/09/2003 11:04:23 Removing CURRENT file for serverb...
WARN 21/09/2003 11:04:23 Remote server not responding - can not remove CURRENT files.
INFO 21/09/2003 11:04:23 Application apache shutdown successfully.
```

22.4.4 The "getmdlist" Utility

This utility can be found in the directory "/sbin/cluster/utlils" and is used on a node that is currently running an application to get RAID information regarding that application. The most useful information is returned when using the "--status" option, as shown below:

```
servera# /sbin/cluster/utlils/getmdlist -application apache -status
app01vg.admin:0:18:active:local:raid1:syncing:18060/20480:166.000000:0.200000
app01vg.logs:10:2:active:local:raid1:unsync
app01vg.docs:1:19:active:local:raid1:unsync
```

23 Implementation Details of “clrunapp”

23.1 Introduction

The previous chapter included lots of information covering the decisions that the “clstartapp” utility made in ensuring that when an attempt to start an application on a node was made that the specified node was suitable and access to a valid copy of data was available.

The “clstartapp” was always designed to be a “low-level” utility – that was to provide a powerful tool to start applications, (even allowing applications to be started when the local node was not part of the cluster). However it has several “limitations”:

- *It is too powerful* – the ability to start applications when the cluster is not running, or against a stale copy of the data is something you really want to discourage users from doing, unless they absolutely must.
- *Complexity* – it supports many command line options – most of the time a simpler interface to get an application started quickly is preferable.
- *Locality* – The command had to be run on the node where you wished to start the application. Not really a problem, but you might have to keep changing nodes when attempting to start lots of applications.

For the above reasons the “clrunapp” command was designed, (and first implemented in version 0.6.0). The interface to the “clrunapp” command is very simple:

```
# clrunapp --application X [--node X] [--nodeps)
```

So in most cases you simply need to pass in the name of the application you wish to start – as simple as that. This command is designed to make use of some additional directives that can be supplied as part of the application configuration file for the cluster:

```
PREFERRED_NODE
```

This directive appearing in the “global” section of the configuration for an application has the following purpose:

This defines the name of the node which should be used to run the application assuming that both nodes are running cluster daemons. Please note that this choice is entirely based on the concept of each node being “UP” - that it is running a cluster daemon. The choice does not take into account (at least currently) of the location of the valid copy of data, but the underlying “clstartapp” - when called – is able to deal with the processes running on one node whilst the data resides on another.

The returned node name will be the value of “PREFERRED_NODE” if set, or the only node in a cluster if the other node is not available. There is also a special value that can be used - LEAST_LOAD - if both nodes are available then node name returned is based on the node currently running the least number of packages.

If not specified the current node or the node with the local data, or the --node command line option will be chosen.


24 Understanding the “Lems” Daemon

24.1 Introduction

This is probably one of the simplest components of the cluster infrastructure as a whole, yet possibly is the most important. Lems - “Linux Event Management System” is a program that is responsible for monitoring an application in the cluster and taking actions based on the results of the monitoring.

At its heart “Lems” could almost be considered as a simple scheduler - at regular intervals it will wake up and run particular monitors. It will then perform actions based on the return codes each monitor. When a “Lems” process for an application is started by the cluster it runs as a background task - and thus it is possible that several different “lems” daemons can be running on the same node at any one time.

Like most of the software currently for “Linuxha” the “lems” program is written in Perl. The major advantage of this approach is that it provides a simple object interface for monitors - allowing additional modules to be easily written. It is also possible to call executables to run though such an approach is less flexible. It does not put a limit on the number of monitors it can run, and the interval can be configured to be anything between .5 and 3600 seconds between monitoring.

 It is recommended that any monitor that needs to be run more often than once every 10 seconds should ideally be written in Perl and should contain no execution of external commands - reducing system overhead.

The purpose of “lems” is to run and respond to monitors at regular intervals - it is not a job scheduler in any classic sense - it does not guarantee what and when or put any ordering of the monitors it runs. It is also single-threaded, meaning that the monitors that do run must relinquish control quickly otherwise other monitors will not be able to run.

24.2 The Lems Configuration File

Lems requires a configuration file to do any useful work. There is one configuration file for each “Lems” daemon currently running. In common with all user-editable configuration files supported by “Linuxha”, this is a basic XML file. An example configuration file can be found on the following page.

The example file contains three entries, and these example entries are based around the requirements for monitoring the network functionality for a particular package. It is envisaged that the actual cluster utilities will be able to generate skeleton configuration files for “Lems” for levels - though currently it is the responsibility of the administrator to take a sample configuration file and manually adapt it to the requirements for the particular application that is being clustered.

```

<?xml version="1.0"?>
<lems_config>
  <globals modules="/sbin/cluster/lems/modules"
            programs="/sbin/cluster/lems/programs"
            logs="/var/log/cluster/lems/logs"
            port="8900"
  />

  <check>
    <name>flag_check</name>
    <type>internal</type>
    <module>flag_check test01</module>
    <interval>5</interval>
    <action_list>
      <action rc="0" action="NOP"/>
      <action rc="1" action="%RCDATA%"/>
      <action rc="2" action="ABORT"/>
    </action_list>
  </check>

  <check>
    <name>httpd</name>
    <type>internal</type>
    <module>procmon /etc/cluster/apache/httpd.xml</module>
    <interval>10</interval>
    <action_list>
      <action rc="0" action="NOP"/>
      <action rc="1" action="STOP"/>
      <action rc="2" action="FAILOVER"/>
    </action_list>
  </check>

  <check>
    <name>ip</name>
    <type>internal</type>
    <module>ip_module test01</module>
    <interval>10</interval>
    <action_list>
      <action rc="0" action="NOP"/>
      <action rc="1" action="RUN move_ip"/>
      <action rc="2" action="STOP"/>
    </action_list>
  </check>

  <check>
    <name>link</name>
    <type>internal</type>
    <module>link_module test01</module>
    <interval>2</interval>
    <action_list>
      <action rc="0" action="NOP"/>
      <action rc="1" action="RUN move_ip"/>
      <action rc="2" action="STOP"/>
    </action_list>
  </check>

  <check>
    <name>move_ip</name>
    <type>internal</type>
    <module>ip_move_interface test01</module>
    <interval>0</interval>
    <action_list>
      <action rc="0" action="STOP"/>
      <action rc="1" action="ABORT"/>
    </action_list>
  </check>

  <check>
    <name>fsmon</name>
    <type>internal</type>
    <module>fsmon apache</module>
    <interval>10</interval>
    <action_list>

```



```

        <action rc="0" action="NOP" />
        <action rc="1" action="PAUSE 30" />
        <action rc="10" action="PAUSE 60" />
        <action rc="2" action="STOP" />
        <action rc="3" action="FAILOVER" />
    </action_list>
</check>
</lems_config>

```

In the above example, the first section is the “globals” section which where logs are written, and where Perl modules to load and external programs to run are expected to be found. This section also contains the port number on which this particular lems service should listen on for requests. For information on the requests understood by the lems daemon see the “Server Messages” section below.

The rest of the configuration file is a series of checks that are run. For each check the following fields are defined:

Element	Purpose
Name	The name of the check - each check must have a unique name.
Type	The type of the check - either “internal” or “program”.
Module	The name of the module or program to use, and any arguments to pass to it. If this is a module there is no need to put on the “.pm” on the end - it will assume this.
Interval	The number of seconds to wait before running the check for this monitor. This is the minimum amount of time to wait. It can be any value, upwards of 0.5 seconds.
Action_list	This is a list of possible return codes from the program or module. If no matching return code is found a warning is issued in the log, but no further action taken.

The available actions are described below.

24.3 The Lems Action List

One of the key aspects of the “Lems” system is that of the action list. For every event to run it is possible to define a list of actions to be carried out based on the return code from the check.

The table below shows the available return codes currently supported:

Action	Purpose
NOP	No operation. Do nothing - “Lems” will continue to schedule this check and will look for further checks to run.
HALT STOP [check]	Indicates that if the specified check is scheduled it should be suspended and not be checked until further notice. If the check is not specified then the current check is suspended.
START RUN check [int]	This indicates that specified check should be started again if it is not currently running. The “int” is optional and specifies the new interval to use when checking.
ABORT FAILOVER	Indicate that the Lems scheduler should abort. The lems module should abort, but before doing so send over a FAILOVER message to the local cldaemon for the current application, causing the application to stop or fail-over, depending on the status of the cluster.
HALTAPP STOPAPP	Rather than fail-over the application simply shut it down and do not attempt to start it on the other node in the cluster.
PAUSE [X]	Do not alter the interval for the specified check, but do not schedule it again for X seconds. If “X” is not specified then it defaults to 30 seconds.
RUNCMD cmd arg..	The specified external command is run along with the supplied arguments. See the notes on this functionality below.
REMOVEMON	When this value is used the monitor in question is completely removed from the configuration.

The “Abort” option is used to indicate a situation where the currently monitored cluster is either being stopped by the user, or a condition requires the node to fail-over.

For each return code more than a single action can be specified - use a semi-colon to separate multiple actions. If any action is not possible or is syntactically incorrect it will be ignored, and any remaining actions will be carried out. For example:

```
RUNCMD email root; rubbish; FAILOVER
```

In the above case the external command would be run, the second command would be ignored and the final “FAILOVER” command would then also run.

When using the “RUN” action the specified check will be scheduled to run at the current time plus the interval, rather than immediately.

In the example configuration on the previous page one of the action list entries for the “flag_check” monitor was “%RCDATA%”. When this is used then monitor is able to set the \$::RCDATA variable to a value which is then used as a replacement for the “%RCDATA%” occurrence in the action. This allows the actions list to be dynamic.

24.3.1 Using the "RUNCMD" Action

When this action is used the specified external command is run. All standard output and standard error text are redirected into the following files respectively:

```
/var/log/cluster/lems/application-monitor.stdout  
/var/log/cluster/lems/application-monitor.stderr
```

The command in question is run in the background - that is if multiple actions are specified do not rely on the external commands from completing first.

Currently there is no way of capturing the return code from the external command and configuring lems to act on this value. If necessary the external command, (which would be a simple shell script for example) should use the "lemsctl" command to send messages to the daemon.

24.4 Standard "Lems" Monitors

As previously stated the monitors used can either be classified as "system" or "application". "System" monitors are pre-built and are used to manage the cluster software and configuration. Not all system monitors need be used - the configuration of the monitors is currently in the hands of the implementer, though several templates will likely be included.

If a "Lems" monitor is a binary program/ script rather than a Perl package, then it should be located in the directory `/sbin/cluster/lems/programs`. The Perl packages should be located in the `/sbin/cluster/lems/modules` directory.

The example of a "Lems" configuration file previously showed some of the monitor scripts that will be typically included:

Monitor	Purpose
flag_check	This checks the directory <code>/etc/cluster/<application>/flags</code> for files and when it finds one matching a name of another lems object it will halt those checks. Removing files matching objects will cause those checks to be re-scheduled again.
ip_module	This is used to monitor the specified applications IP address. If it is unable to find the address, or does not get the number of responses expected it will send a return value indicating the IP address should be moved to another interface.
link_module	This is a low level interface check. It will find the interface used by the application IP address and if supported will check that the card has a link to the switch. If supported and it does not, then it will set a return code to move the IP address to another interface.
ip_move_interface	This is the module that is included but not scheduled to run. It is triggered by instances of the "ip_module" or the "link_module" and will attempt to move the current IP address to another interface. If it believes that it has been called too often or no other interfaces exist it will abort, causing the "Lems" session to exit with a failover message – which is sent to the local cldaemon to initiate a fail-over of this package.
fsmon	This is probably the most important monitor that exists – it is responsible for handling data consistency between the local and remote copies of data. Much more information on this module can be found below.
procmon	This is a general process monitor and one or more instances of this module are typically used to monitor the application processes.
swap_check	A monitor that can be used to trigger actions if the available free swap space drops below a specified minimum.
capacity_check	Checks the free space in one or more file systems, and if the used capacity exceeds a threshold an action is taken.

Each of the standard checks above will now be discussed in a little more detail.

24.4.1 The "Flag_check" Module

This module in essence is simply used to stop and start other modules! Although this sound a bit strange it is actually a fundamental requirement for high availability software. Typically this might be used to stop monitoring of part of the application whilst some changes take place - otherwise the monitoring may cause the application to be re-started, or attempt a fail-over.

It will send "STOP" messages to the module named as a file in the application directory on the local machine, which will be `/etc/cluster/<application>/flags`. If the monitor specified does not exist a warning will be given.

When the specified file is removed from the directory the module will send the target module a "START" message to ensure the module is started again.

As can be seen from the sample "Lems" configuration file, all actions are handled by a return code of 1, setting `$_RCDATA` to store details of modules to start and stop at this moment in time.

24.4.2 The "Ip_module" Module

This module is responsible for checking IP connectivity to those IP addresses defined for this package, (if any). It also ensures that one of the interfaces specified actually has the IP address for the package assigned to it – and if not assigns it.

When this module is unable ping the addresses successfully it will exit with a return code of 1, which is used by the Lems session to call the "ip_move_interface" routine.

24.4.3 The "Link_module" Module

This module checks for link level activity on the network interface currently in use for this package – if the specified interface supports such checks. If link beat can not be established it again uses return code of 1 to ensure Lems activates the "ip_move_interface" module.

24.4.4 The "Ip_move_interface" Module

This will move or assign the IP address for the current package. It maintains a history of activity – and if a certain interface has already been used recently it is discarded from the list of candidate interfaces to use.

If the return code is 0, it means that the IP address has been assigned to an interface and this module can be put to sleep. If 1 is returned, it indicates to "Lems" that the "Failover" action should occur – which may result in a fail-over for the package being attempted.

24.4.5 The “capacity_check” Module

This can be useful if a particular application relies on one or more local file systems having space available for it to operate - such as in “/var” and “/tmp”. The file systems to monitor are specified using the following syntax:

```
FS:CAP[ , ... ]
```

For example to trigger if “1” return code on “/tmp” being greater 75% full and/or “/var” being 90% full, the argument specified to the monitor would be:

```
/tmp:75,/var:90
```

This module requires that the “syscall.ph” file is available, since it queries the available free space via the “statfs” system call.

If the monitor fails the header files may need to be converted to suitable Perl equivalents using a command such as:

```
# cd /usr/include; h2ph -r -l .
```

Please consult the Perl documentation for more information on the “h2ph” command.

24.4.6 The “swap_check” Module

This simple monitor checks the available free swap space and if it drops below a certain threshold will return a “1”. The threshold is specified in Kb - and if not specified will default to 51200 - 50Mb.

24.4.7 The “Fsmon” Module

This module is responsible for ensuring that the local and remote copies of the data for a certain application remain in sync if possible, and if not possible, are flagged as being out of sync as soon as possible.

When a particular logical volume becomes out of sync it will attempt to schedule re-synchronisation if possible, for each affected volume, clearing the out-of-sync flag once the synchronisation is completed.

If necessary it will stop/start some or all of the NBD or ENBD clients and servers to ensure that all possible avenues to start synchronisation are performed.

This module interfaces with the local cluster daemon to perform any actions on the cluster. This ensures that the cluster daemon contains all the necessary functionality, whilst this monitor remains free of as much of the resynchronisation code as possible.

Since the recovery mechanisms are different if the “fr1” module is in use compared to the “raid1” personality the code checks the type of RAID in use and alters the behaviour it uses accordingly.

The following table shows the various state changes that the monitor uses to assess the current conditions of the environment.

State	Checks / Purpose
0	Original status – here we ascertain the status of the RAID device information to see if any devices are currently STALE. If any stale devices are found the information is sent to the local cldaemon (if running) via the following message: <pre>FLAG_UNSYNC app=X vg=Vg lv=LV invalid=local remote forward=yes</pre>
	Any synchronised devices send the following message to the local daemon:

State	Checks / Purpose
	<pre>FLAG_SYNC app=X vg=Vg lv=Lv forward=yes</pre> <p>If no valid response occurs then the State is changed to 10, Substate to 0, which is used to indicate that it appears that the daemon is currently down. If the flags have been sent successfully we set the STATE to 10, substate 11 – which is used to check NBD IP communication is available, and create a new STALE_LIST and set the current syncing device (MD_SYNC) to the undefined value.</p>
	<p>Firstly there is a check to see if communication channels exist to the local and remote daemons, and if they do not then the state is changed to 10, using sub-states 0 and 1 respectively to ensure that this communication is available.</p> <p>Now the substate is defaulted to 1 and “tries” to 1 if substate is not currently set, then the following substate checks occur:</p> <ol style="list-style-type: none"> 1) The following message is sent to the remote daemon to ensure that the ENBD server processes are running: <pre>CHECK_NBD type=server app=App</pre> <p>If the response is “OK” then substate is set to 2 and this check exits. If the check is not “OK” then the following message is also sent to the server: <pre>START_NBD app=App</pre> </p> 2) If the response is OK then the substate is changed to 2 and “dorefresh” is set. Otherwise the routine will try again for a maximum of 3 attempts before pausing and waking up 60 seconds later at try everything again. 2) The local daemon is sent the following message to check the NBD client connections: <pre>CHECK_NBD type=client app=App</pre> <p>If the clients are OK then the state is set to 9, and “dorefresh” is 0. If the result is not OK then the following message is sent to attempt to start the client side processes: <pre>START_NBD app=App</pre> <p>If is attempted a maximum of three times, before resetting the State to 0 again after a pause. If this works then the State is again set to 3 and “dorefresh” is set to 0.</p> </p>
	<ol style="list-style-type: none"> 3) If no devices exist in the “stale_list” then return to state 0. Otherwise is any NBD devices are marked as “disabled” a “0” is sent to “/proc/nbdinfo” to attempt a recycle of the devices. If this has occurred for three times then a warning is issued and state is returned to 0. <p>If no ENBD devices are disabled then a “raidhotadd” is sent to all devices, (additionally a “raidhotremove” is sent if “fr1” is not detected). After this the state is changed to 6, and “insync” is set to an undefined value, and “tries” is reset to 0.</p>
	<p>This state is used when we expected devices to be synchronised. It will check to see if any “insync” device has been set and if not, will set one, if one is available, (device that is currently in progress of being synchronised).</p> <p>If no stale devices are found, (remote stale devices more accurately), then the state is return to 0 – which then sends the SYNC status information to the daemons.</p> <p>If no devices start to sync after 3 tries then a warning is issued and state is return to 0 to look for other problems.</p> <p>If a device is being synchronised then the number of blocks to sync is checked. If this stays the same for 3 passes then state is returned to 0 for further checks.</p>
10	<p>This state is used to indicate that the local daemon does not appear to be running (typically if the server is not actually running, or there has been a time-out due to other cluster activity).</p> <p>If the Substate is 0 then it attempts to create a new channel to the local daemon and if it succeeds then it keeps State as 10, but sets the Substate as 1.</p>

State	Checks / Purpose
	<p>if it fails then it sets a return code of 1 indicating that the test should be repeated again in 60 seconds.</p> <p>When the substate is 1 then the purpose is to validate that a remote connection is available, and that the remote daemon can be contacted. Again if it can not then a pause of 60 seconds is expected due to a return code of 10.</p> <p>If the substate has been set to 11 then aim is to validate that communication can be attempted over the NBD channel and if not to try again after 30 seconds. Once the communication has been established then the state is returned to <code>SAVED_STATE</code> or 0 if not defined.</p>

24.4.8 The "procmon" Modules

This module is used to monitor processes. The argument given is the name of a configuration file to parse which contains the information to monitor. The return codes can indicate that no action should be taken, (normal conditions), that the monitor should stop (a non-fatal, probably coding error, has occurred), or initiate a fail-over, since the application being monitoring is not running, or can not be restarted.

A sample configuration file might look like the following:

```
<?xml version="1.0"?>
<procmon>
  <global>
    <logdir>/var/log/cluster</logdir>
    <restarts>3</restarts>
    <resetwindow>3600</resetwindow>
    <restartcmd>/apache/admin/apache restart</restartcmd>
  </global>
  <process>
    <label>Web Server</label>
    <user>nobody</user>
    <process_string>httpd</process_string>
    <min_count>1</min_count>
    <max_count>10</max_count>
  </process>
</procmon>
```

In the above example, the "process" entry can be repeated several times, allowing the same module to monitor different processes. A summary of the entries included in this configuration follows:

Field	Purpose
global.logdir	The directory where log files are kept for the monitor in question. If this is not specified it will default to "/var/cluster/log". The actual name of the log file will be in the following format: procmon-<application>-<name>.log
global.restarts	The number of times the process should attempt to be restarted on this host before considering a fail-over to the other node in the cluster. Must be specified.
global.restartcmd	The command to run to restart the application that is being monitored. Should be the full pathname and can refer to the paths provided by the package if necessary.
global.resetwindow	The length of time after which the current number of available resets is set back to the default (in seconds). If this is not specified it defaults to 3600 (one hour).
process.label	The text string to use in the log file if the specified process fails. Must be present.
process.user	The user to look for processes against, if not specified will default to "." - which will match all users.
process.process_string	The process string to look for - must be present. This can be a normal string or a Perl regular expression.
process.min_count	The minimum number of instances of the process that must be running, default is 1.
process.max_count	The maximum number of instances of the process that must be running, default is 999.

It also accepts the following custom commands to manage the status of the monitor (for information on how these messages are sent please see the following section).

Message	Purpose
RESET	Reset the restart counter to 0.
MAXCOUNT N	Change the maximum number of restarts before triggering a

Message	Purpose
	fail-over to "N".

24.5 The "Lems" Server Messages

Each instance of a lems server has the ability to listen on a port specified by the "port" attribute in the global configuration options. This encrypted channel uses the same key for the cluster as other communication channels and is used primarily to ensure the "clstat" utility, the cluster daemon "cldaemon" and "lems" daemons can communicate where necessary.

At the moment the range of requests accepted and actions by the lems server are quite limited as the following table below shows.

Message	Purpose
GET_STAT [monitor]	<p>This will return the status of all monitors for the given session. The output will appear in the following format:</p> <pre>Event,nextrun,interval,command<nr></pre> <p>...</p> <p>If the monitor is specified then just a line for the that monitor is shown, otherwise the detail for all monitors is shown. "nextrun" is the UNIXTIME in the future when the monitor is next scheduled to run. 0 means that the monitor is currently stopped.</p>
GET_VSTAT [monitor]	<p>This is similar to the GET_STAT routine except that after the status line for a particular monitor the output will include:</p> <pre><monitor-detail-start> detailed monitor information <monitor-detail-end></pre> <p>Note that the "detailed monitor information" may be 0 or more lines of text – the format of which is monitor independent.</p> <p>Again the routine can include a monitor name to restrict output just to that monitor.</p>
CMD monitor msg	<p>This sends the specified text to the specified named monitor if it exists. It will return "OK text" if the message was sent successfully, "NOT_IMP" if the monitor does not implement the "accept" method, or "FAIL text" if the monitor indicates this has failed.</p>
PAUSE [monitor]	<p>Stop ALL monitoring until further notice - typically used during periods of cluster reconfiguration. If the "monitor" argument is given then just the specified monitor is stopped (if it exists).</p>
RESUME [monitor]	<p>Resume monitoring as it was prior to a PAUSE request. If the "monitor" argument is given just the specified monitor is resumed.</p>
REMOVE monitor	<p>Removes the specified monitor from the configuration – obviously use with care!</p>
INSTALL monitor	<p>Adds the specified monitor name from the configuration – this monitor must not already exist. The object type code is also re-sourced for this monitor meaning that the underlying module code is also changed dynamically – good for maintenance.</p>
ABORT	<p>The lems session is aborted immediately - though the application will continue to run without monitoring.</p>
VERBOSE ON OFF	<p>Allows the verbosity of the daemon logging to be altered without the daemon having to be restarted.</p>
LOGCYCLE NN	<p>Forces the daemon to switch log to a new log. The argument is the number of older logs to keep. The older log names use the following naming scheme:</p> <pre>logfile_name.NNN</pre> <p>The first file has an extension of "000".</p>

All commands are sent to the daemon using the "lemsctl" command, which has the following syntax:

```
# lemsctl --application appname --msg "message"
```

24.6 The "Lems" Module Object Method Requirements

The "Internal" type of check is recommended when authoring new modules for the following reasons:

- Efficiency - since the modules actually form part of the main "Lems" program, the amount of effort to run the check is greatly reduced, (i.e. no fork/exec requirements).

- Stateful - since the same object is used for every occurrence of the check the object is able to store state information - this is used by the "ip_move_interface" module for example to ensure we attempt a fail-over if Ip address moves are occurring too often.

The Object must support the following methods:

- new(arguments)

The arguments passed to new are those given in the "module" element in the configuration file. White space is used to separate the arguments.

It is expected that a call to this will return a blessed reference or "undefined" if the object can not be created. In the case when an object can not be created it is obviously removed from the run schedule.

- check

Notice that no arguments are passed to this method. This method must use the information passed to it from the new method, or the environment (see below), to perform the required check. It is expected to return a numeric value which will then be used by "Lems" to work out the action to perform, if any.

The check can also set the \$::RCDATA value to be substituted in the chosen action for the string "%RCDATA%".

The following methods are optional:

- stat

This is an optional method which is called when the lems monitor receives a GET_VSTAT request. If not specified the request will return empty information to the requester.

- accept

This is another optional argument allowing the monitor to accept a "CMD" request issued to the client environment. This is typically used in application monitor entries to do things like reset counters.

24.6.1 Object Environment

Apart from the \$::RCDATA variable, the following subroutines are available to the "new" and "check" methods:

Subroutine	Purpose
::logwarn(msg)	Log a warning message to the console or log file.
::logmsg(msg)	Log an informational message (if verbosity has been chosen), to the console or log file.
::errmsg(msg, rc)	Abort the lems session with the specified error message and return code.

The objects also have access to the following global variables:

Variable	Purpose
\$::RCDATA	Used for a substitution value for the %RCDATA% data macro in the action list.

<code>%::SCHEDULER</code>	Hash where each key is the name of a check from the current configuration file. Each value is a reference to a hash containing the "EST" and "OBJECT" keys, containing the estimated next run time and the OBJECT reference.
---------------------------	--

24.7 The "Lems" Program Execution Requirements

To be written

24.8 Writing a Sample Lems Monitor

The purpose of this section is to document how a simple Lems monitor can be written by using the "swap_check" monitor as an example.

24.8.1 The "new" method

Since this monitor is very straightforward the "new" method only sets up a "LIMIT" variable to remember the limit to check for:

```
package swap_check;

sub new {
my $self={};
my $lim;

    if(exists($_[1]) && defined($_[1])) {
        $lim=$_[1];
    } else {
        $lim=51200;
    }

    $self->{LIMIT}=$lim;
    bless($self);
    return $self;
}
```

24.8.2 The "check" Method

The check method in this case is very straightforward - it simply queries the "/proc/meminfo" file and returns "0" if enough free swap space is available or "1" if not.

```
sub check {
my $self=shift;
my ($fd);

#####
# Read in the memory information extracting required field #
# from the line: #
# SwapFree:      NNN Kb #
#####

open($fd,"/proc/meminfo") or return(0);
while(<$fd>) {
    next if ! /^SwapFree:\s+([0-9]+)/;
    $self->{LAST}=$1;
    close($fd);
    if($1 < $self->{LIMIT}) {
        return 1;
    } else {
        return 0;
    }
}
close($fd);
return 0;
}
```

Please remember that since the code is part of an object that persists you must ensure that all file descriptors are closed before returning from the "check" method, unless you specifically define the check to be stateful and use the open file on subsequent calls.

24.8.3 The "stat" Method

The "stat" and "accept" methods are optional, and can return any string value. In this instance an empty string is returned or the available free space and configured threshold configured comma separated if the "check" method has been called.

```
sub stat {
my $self=shift;

    if(exists($self->{LAST})) {
        return $self->{LAST} . "," . $self->{LIMIT};
    }
    return "";
}
1;
```


25 Understanding the Cluster Management Daemon

25.1 Introduction

Although previous information in this document covers what this daemon does, and how it can be started, this chapter intends to cover the architecture of the daemon in some detail, and describe the functionality it offers, who it is implemented, and why.

The cluster management daemon is responsible for co-ordinating all system-related cluster reconfiguration operations. This means that it is the cluster management daemon, which takes notice of the failure of the “Lems” session for an application and initiates a fail-over to the other node.

Notice that the “Lems” sessions and the Cluster Management daemon communicate using sockets on the local machine, though it is expected that all communications will be encrypted and check summed using the key for each package to ensure that only those knowing the key for a package, (which should be the “root” user only), is able to send valid messages to the daemon.

 Although the current version of the architecture does not include these daemons “voting” on the status of the cluster, such functionality may be used in the future (when clusters will be larger than two nodes).

25.2 How the Cluster Daemon Interacts with an Application

Firstly it should be noted that it is possible to start an application without the cluster services running - however this is not recommended for the following reason:

The cluster daemon is responsible for handling fail-over events and without the daemon running this will not happen!

When starting an application the last action will be to check that the cluster daemon is running and if so, send it a “STARTED_APP” message indicating that an application has been started on the local node, and it will need to use “Lems” to monitor it and respond as appropriate.

If the cluster daemon is not running the “clrunapp” function will simply fork/exec the “Lems” daemon in “daemon” mode - causing it to fork and then use “setsid()” to detach from the current session. This is achieved by using the “--detach” option when invoking Lems.

The Cluster daemon can be communicated to via a well known socket - this is fixed and appears in the cluster configuration file, as the “port” option. All messages are of a certain format and are encrypted, (see the information starting on page below). The “cldaemonctl” utility provides a simple interface for sending messages and seeing the response from the cluster.

Thus the Lems session can attempt to signal a fail-over, though unlike an application such as “MC/ServiceGuard” having the monitor simply die is not enough – it will need to send a message to the local daemon to explicitly indicate that a fail-over is desired. This ensures that if a bug occurs in the “Lems” system, (which is distinctly possible given that administrators can add or alter modules if they desire), this will not cause a fail-over event to occur.

However, “Lems” is not the only command that interacts with the cluster daemon – the “clstat” command communicates with a running daemon (ideally locally), and also the Lems session for an application if detailed information for a particular application has been requested.

Further the commands “clstartapp” and “clhaltapp” also send messages to the daemon to indicate that the status of a package is changing, (indicating it is stopping, starting, stopped or running).

25.3 Forming the Cluster - Cluster Daemon Initialisation

This section describes the actions taken by the cluster daemon when it is started - via the "clform" command rather than the "cldaemon" command directly.

The "clform" command accepts the following command line arguments:

Argument	Purpose
--noapps	Do not start any applicatoins that typically would be automatically started.
--force	If the cluster daemon is unable to find start / find the details of the other node it still starts (otherwise would fail).
--join	Indicate that the cluster is already running and the node not currently running should join the cluster.
--config	Specify an alternative configuration file to look at for the cluster configuration - typically used for development of the product only.

Notice that the "--force" functionality allows the cluster to form with only a single node being present - or with both nodes present even if they do not have close or identical clocks (which is obviously not recommended).

The "clform" command is responsible for calling the "cldaemon" command on one or both nodes to ensure that the cluster is started or joined accordingly.

When called the "clform" process starts it performs the following checks to ensure that a cluster can be formed or joined.

- Is there a valid cluster configuration file and will necessary mandatory details to start the cluster?

If the cluster is to be performed the following checks are then carried out:

- Check to see that the cldaemon port is not being used locally already (if wishing to form the cluster) - if so abort with an error message.
- The same check is then carried out on the remote node to ensure the cluster daemon is not already running there either.

If the cluster is thought to be running and the "--join" flag has been specified then the following checks are carried out instead:

- Do we believe that the other node is actually "up" - this attempts to get a connection the daemon on the remote node or the local node, and checks that only a single node is actually running.
- The attempt will abort if either both nodes are running, (form not necessary), or neither node is actually running.

At this point "clform" will attempt to run the daemons on either or both nodes in the cluster, depending on whether the cluster is being formed or just a single node is joining the cluster.

The actual process of forming the cluster or joining an existing cluster is then down to the communication the "cldaemon" attempts:

- Either or both nodes have the cldaemon program run with either the "--form" or the "--join" argument depending on the action required.
- If using the "--form" option the following actions take place:
 - ◆ If running the primary server then we send a series of FORMING, STARTING and UP messages to the other node to ensure it is also forming the cluster.
 - ◆ If running on the secondary server then we immediately enter the main loop, but only accept requests to form the cluster. Once formed the series of requests to accept becomes the normal list.
 - ◆ If the time difference between the two nodes is greater than a minute, but less than 10 minutes, issue a warning. If it is greater than 10 minutes then abort cluster formation, unless the "--force" option has been specified.

- When running the with "--join" option the cldaemon on the node that is to join the cluster does the following:
 - ◆ Attempt to get a ping to the other node, otherwise joining the cluster is abandoned.
 - ◆ A connection to the already running daemon on the other node is opened and "ECHO HELLO=yes" message is sent repeatedly using a random delay until a response is returned.
 - ◆ If a valid response is received ("UP" - indicating the other node is running a cluster) prior to the time-out then the cluster is considered joined, otherwise this node aborts joining the cluster.
- The primary server now checks the status of each application configuration file, and if both nodes have the same information each application is registered in a state of "STOPPED".
- Otherwise if the IP address of the application is found then the application is registered as "STARTED", and the Lems daemon for that application is probed to see if it is running.

Once the "cldaemon" processes have formed the cluster then the "clstart" process will exit.

25.4 Protocol and Messages

Since the cluster daemon is responsible for reporting on the status of the cluster and applications, as well as automating fail-over, many messages are passed to it from user-invoked actions, or between itself and the remote node to ensure both retain consistent state information.

The general format of any message/request sent to the daemon is as follows:

```
MD5,REQUEST arg=val...
```

The complete request should be encrypted using blowfish using 448 bit encryption using the codeword defined in the cluster configuration file. The "MD5" is the hex value of the checksum of the unencrypted message to ensure there has been no corruption. The response from the server back to the client will be in the same format.

The request and response are terminated by a line feed. Thus the encrypted string is actually passed as hex rather than binary to avoid any confusion.

The requests handled by a cluster daemon are as follows. Please note that some of these requests are sent from one daemon to another to ensure consistent cluster state is kept. In the above table, yellow indicates something has been tested and currently works, whilst dark grey means that it has not yet been written. The "cldaemonctl" program can be used to send the daemon messages – as explained in previous sections of the documentation.

Request	Purpose
TIME	Returns the current time on the node (UNIX time). This is used by the primary to check that the nodes are within a certain time tolerance when starting, or when a node joins the cluster.
ABORT	When this is received the cldaemon in question will exit, no matter what the status.
GET_NODE_STATE node=node	Will return the status of the specified node in the cluster, which can either be the node on the same node, or the other node.
APP_UPDATE app=name state=STATE [node=node Monitor=monitor starttime=time consistency] probe=yes [forward=yes]	This is used by one daemon to tell the other that the specified application state has been changed. States are STOPPED, STOPPING, STARTING and STARTED. All other field information is optional. If the PROBE option is specified it attempts to see if the IP address for the application is currently held by this machine, and if so sets the application to UP status, and informs the other node. Forward is only used testing outside of the daemons...
STARTED_APP app=name	This is sent from the clrunapp invoked by the user to tell the daemon that the specified application has been started. The daemon will be responsible for starting the required lems session if the configuration file exists. Will trigger an APP_UPDATE message to the other node in the cluster, (if it is available).
STARTING_APP app=name node=node	Message sent from one daemon to another to ensure that the other daemon knows a particular application is starting on the other node. Will trigger a APP_UPDATE to the other node in the cluster. Will respond with OK, ALREADY_STARTING, ALREADY_STARTED or NOT_REGISTERED depending on whether this application is ok, currently being started, is running or is unknown. If the node is not valid for this application, (due to a previous

Request	Purpose
	software failure), it could also return "INVALID_NODE" - introduced for version 0.6.0 upwards.
START_APP app=name	Indicates to a node it should start the specified application using the clrunapp itself. It should then send the remote node a STARTING_APP message.
STOP_APP app=name	This will stop the specified application on the local node. This is typically used by lems to halt an application if it believes that a fail-over is not a suitable action. This will only work on the node that is currently running the application, so ensure it is sent to the correct node. Return codes will include, "OK", "NOT_RUNNING" and "NOT_REGISTERED".
STOPPING_APP app=name	Indicates to the remote node that the specified application is currently in the process of stopping, (and so monitoring of remote services should stop).
STOPPED_APP app=name	Indicates to the remote node that the specified application has now completed stopping and is down.
ECHO [check=y]	Sent by one node to another to check if the specified daemon is up and running. When the "check" argument is specified the node does not use this information to validate its partner is available, (used by clstat command).
CHANGE_DSTATE state=state	Change the status of the specified node - used when a cluster daemon is starting.
ABORT [forward=yes]	This forces the daemon to abort - typically done when the cluster is starting or stopping.
CHECKSUM app=name	This will the contents of the build.md5 file for the specified application (if it exists). This is used by the cluster to ascertain whether the contents of the build are the same and valid on both nodes. If it does not return the MD5 of the file then either of the following is returned: undef – remote node not responding MISSING_ARG – missing the APP argument NO_SUCH_APP - specified application does not exist. NO_SUCH_FILE – the specified md5 file does not exist. INVALID_FILE - the file exists, but does not appear to include any data.
APP_LIST	Will return a comma separated list of applications that the cluster daemon currently knows about.
APP_STATUS app=name [lvinfo=yes] [failoverinfo=yes] [nodeinfo=yes]	Get the status, node, start time, monitor (lems running or not), and node consistency of specified application. The response is in the following format (comma separated): STATE - (STARTED,STARTING,STOPPING,DOWN), server - on which server it is monitor – 1 if lems monitor running, 0 otherwise start time – the time package was started consistency – number of inconsistent file systems if "failoverinfo" is passed an additional element is returned – "Yes" or "No". This indicates whether the application will fail-across. It will say "No" if the node is not available, or the valid list for this application does not include the other node. If "nodeinfo" is passed then an additional argument is returned – the contents of the "validnode" list for this application, with "+" is a separator between node names. If "lvinfo" is passed then it will return the error state or "OK" on

Request	Purpose
	<p>the first line, and then the following fields for each LV, newline separated:</p> <ul style="list-style-type: none"> • vg.lv – The volume group/logical volumes • MD – The RAID device number • ND – The NBD device number, (must be mapped to character name if required). • State – active or stopped or inactive. • Valid – Valid data source local, remote, or both • Personality – raid1 or fr1 • Action – Sync , unsync or syncing <p>If the action is “syncing” then the following fields will also be present:</p> <ul style="list-style-type: none"> • valid/total – Kb valid and total blocks for this devices • Rate – Rate of synchronisation in Kb/sec (floating point). • Estimation – The estimated completion time in minutes (floating point).
CHECK_NBD app=name [type=client server]	<p>Given the specified application this will check to see if the specified nbd clients (if this is the node running the application), or nbd servers (if this is NOT the node running the application), are in fact running.</p> <p>The return values are as follows: OK – All servers or clients for the application on this node are running. NOT_OK – One or more clients or servers on this node are not running. MISSING_ARG – Missing the app argument. MISSING_CMD – Unable to find the nbd_svstart or nbd_clientstart commands.</p>
STOP_NBD app=name [forward=yes] [type=client server] [vg=vg lv=lv]	<p>This will stop the client and server for the specified NBD or ENBD daemon for just the specified logical volume. If you send it with forward it will send the request on to the other node.</p> <p>The cldaemon knows whether to stop the clients or servers, depending on whether the application is running on that host, though the optional type can be used if you wish to be explicit.</p> <p>The “VG” and “LV” entries are optional – if they are not specified then ALL entries for the application are stopped.</p> <p>The return results are “OK”, “FAIL-LOCAL”, “FAIL-REMOTE”, “MISSING_ARG” and “NOT_REGISTERED”.</p> <p>Note: If the “type=” setting is used then it will forward the request to the other node to answer the query if the application is currently running, and the other daemon is available.</p>
START_NBD app=name [forward=yes] [type=client server] [vg=vg lv=lv]	<p>This will attempt to start the specified NBD or ENBD server and client on the necessary names for the specified NBD service only.</p> <p>As above the “LV” and “VG” are optional and if not specified all entries will be started for the specified application, (or restarted if they are already running).</p> <p>This is typically used prior to a synchronisation attempt to ensure the daemons are running. The return codes are as above.</p>
FLAG_UNSYNC app=name	This is sent to indicate that flag files should be created to

Request	Purpose
vg=vgname lv=lvname invalid=local remote [forward=yes]	<p>indicate the specified logical volume has synchronisation problems. It will create the flags locally and attempt to create them on the remote node if possible.</p> <p>The return results are as above; "OK", "MISSING_ARG", "NOT_REGISTERED", "FAIL-LOCAL" and "FAIL-REMOTE".</p> <p>This will create the VG.LV.node.STALE flag on the local node, (on both hosts if the "forward" flag has been specified). These flags are created in the following directory: <i>/cluster/control/.status/application</i></p>
FLAG_SYNC app=name vg=vgname lv=lvname [forward=yes] [valid=local remote]	<p>This indicates that the specified logical volume for the specified application is now valid, and hence any flags should be removed from the local and remote nodes if possible.</p> <p>If the ACTIVE value is specified then then just the flags for that node are removed. If FORWARD is specified then attempt to communicate the fact to the other node.</p> <p>The return results are as above; "OK", "MISSING_ARG", "NOT_REGISTERED", "FAIL-LOCAL" and "FAIL-REMOTE".</p>
START_SYNC app=name vg=vgname lv=lvname [attempt=N]	<p>This will instruct the daemon to attempt to get the specified device to be synchronised. It will expect the flags to exist locally to indicate which device is invalid and hence what the direction of the synchronisation.</p> <p>Please note that an "OK" return code does not mean that synchronisation is underway, only that the commands to start a synchronisation have been successful.</p> <p>The "attempt" attribute is optionally and is used to allow the same vg/lv to attempt synchronisation using different methods. The default is 0, but the caller might invoke with higher numbers, until "NO_METHODS" is returned to indicate no further methods to attempt synchronisation exist.</p> <p>Other return codes include "OK", "FAIL-LOCAL", "FAIL-REMOTE" (the local node will forward the request to the remote node if that node currently is running the specified application).</p> <p>Return codes such as "MISSING_ARG" or "NOT_REGISTERED" are also possible.</p>
FAILOVER app=name	<p>This is sent to the local node, and will indicate that the specified package should be failed to another node if possible.</p>
DEREG_APP app=name [forward=yes]	<p>If the specified application is currently registered then de-register it! It will return either "OK", "NOT_REGISTERED" or "MISSING_ARG".</p> <p>If the "forward" argument is given it will send the message on to the other server in the cluster.</p>
REG_APP app=name [forward=yes]	<p>Register a new application with the cluster - it will force the daemon to attempt to ensure that the application is valid and load the details. It will return either "MISSING_ARG", "MISSING_FILE", "NOT_CONFIGURED" or "OK".</p> <p>If the "forward" argument is given it will send the message on to the other server in the cluster.</p>
PNODE app=name	<p>This returns the "preferred" node to run the specified application. This was introduced in version 0.6.0 to provide</p>

Request	Purpose
	improved handling of application start-up – especially when a node has many applications. Return values are “MISSING_ARG”, “INTERNAL_ERROR” or “OK”. The format of the OK message is: OK servername
DEPENDS_ON app=name	This will return a comma separated list of applications that are currently registered that must be started ideally before attempting to start this application. This information is used by the cform and clrunapp to define dependencies. Return values are “OK applist or ERROR_NO_APP”.
AUTOSTARTLIST	This will return a list of applications that are currently registered and have the “autostart” configuration values set to “true” or “yes”. Return values are: OK app1[,app2...)
VERBOSE ON=true OFF=true [forward=yes]	This will turn on or off the verbosity for the local daemon . It will accept the forward option to send on the request to the other node in the cluster as well.
LOGCYCLE count=N [forward=yes]	Cycles the current log (if the log file is a real file). Keeps a maximum of N rotated logs, where N is 1 to 99.
GETVALIDNODES app=name	Returns “OK nodelist”, where nodelist is a comma separated list of nodes that are considered suitable to run the application.
SETVALIDNODES app=name nodes=A[,B] [forward=yes]	This will set the valid list of nodes that are registered to run an application.
ISVALID app=name	When sent to a cldaemon it will respond with “YES” or “NO” or “NOT_REGISTERED”. This will be used by cldaemon to check to see whether the application should be run on this node or not.
HELP	Produces a comma separated list of commands that the daemon is currently responding to.
TOC	The node that receives this signal will reset immediately - without evening performing a shut-down or disk synchronisation.
LOADAVERAGE	Returns the 1 minute, 5 minute, 15minute load average information for the node in the following format: 1min 5min 15min running/procs If there is a problem getting the information it will instead return: ERROR

The list of supported messages is likely to continue to change, and so relying or using a particular message apart from those discussed in cluster administrator steps earlier in the document is not recommended.

26 Cluster Utility Scripts

26.1 Starting "nbd" or "enbd" Servers

The utility `/sbin/cluster/utis/nbd_svrstart` can be used to start, restart or check that all "nbd" or "enbd" servers for a particular application are running on the local machine. The command accepts the following command line arguments:

Argument	Purpose
<code>--application XXX</code>	Indicates the name of the application to run against - must already have been defined in the cluster using the "clbuildapp" "--build" functionality.
<code>--check</code>	Return 0 if all servers are running, or 10 if one or more servers for the application are not running. When servers for an application are not running a space separated list of servers needed to start is shown on standard output, (see examples below).
<code>--newonly</code>	Start servers for the application, ignoring servers that are already running for this application.
<code>--force</code>	Start all servers for the application. If any servers are currently running they are killed off first, (see examples below).
<code>--verbose</code>	Produce additional log messages to the screen.

For example to check to see which servers need starting to service all file systems relating to the "test01" application, please use the following command:

```
# nbd_svrstart --application test01 --check
test01.app01vg.lv1 test.01.app02vg.maps
```

The output elements are in the format of "application.volume group.logical volume".

To enforce all servers to start for an application, killing off any existing servers for the application:

```
# nbd_svrstart --application test01 --force --verbose
```

26.2 Starting "enbd" Clients

Similar to the "nbd_svrstart" utility, the "/sbin/cluster/utls/nbd_clientstart" allows the current server to check, start or restart all client daemons for enbd devices on the current machine.

It will query the remote machine to get details of the ports to run against - thus the remote machine is expected to be up, (otherwise the client "enbd" would not be able to be run anyway).

The utility supports the following command line arguments, (very similar to the "nbd_svrstart" utility):

Argument	Purpose
--application XXX	Indicates the name of the application to run against - must already have been defined in the cluster using the "clbuildapp" "--build" functionality.
--check	Return 0 if all clients are running, or 10 if one or more clients for the application are not running. When clients for an application are not running a space separated list of clients needed to start is shown on standard output, (see examples below).
--newonly	Start clients for the application, ignoring clients that are already running for this application.
--force	Start all clients for the application. If any clients are currently running they are killed off first, (see examples below).
--verbose	Produce additional log messages to the screen.
--local	Treat the "remote" machine as the local machine. This is typically used for testing running both the client and serves on the same machine.

26.3 Stopping "nbd" or "enbd" Clients

26.4 Generating "Raidtab" Configuration Files dynamically

The "/sbin/cluster/utls/mkmdcfg" is a utility to create a "raidtab" file, which is the format of the configuration file used by the "md" RAID software when controlling one or more devices. It is used whenever a given package starts to generate a file containing the configuration of the cluster. It currently takes the following command line arguments:

Argument	Purpose
--application XXX	Indicates the name of the application to run against - must already have been defined in the cluster using the "clbuildapp" "--build" functionality.
--output XXX	The name of the file to generate. This should not include a path element since the utility will always write the files to the correct configuration directory, (see below for an example). If this argument is not supplied it will default to the name of the application.
--volume XXX	Just put the specified logical volume in the configuration file, not all logical volumes for the application. CURRENTLY NOT IMPLEMENTED
--force	If the specified configuration file exists then overwrite it. If the file does exist and this argument is not specified an error will be returned instead.

Typically this utility would be run as follows, (when an application is about to be run from the current machine):


```
/sbin/cluster/utils/mkmdcfg --application test01 --force
```

This would generate a file called "test01" in the following directory:

```
/cluster/control/.resource/mddata
```

This file looks like the following:

```
raiddev /dev/md0
raid-level          1
nr-raid-disks      2
nr-spare-disks     0
chunk-size         4
persistent-superblock 1
device             /dev/app01vg/test
raid-disk          0
device             /dev/nd0
raid-disk          1

raiddev /dev/md10
raid-level          1
nr-raid-disks      2
nr-spare-disks     0
chunk-size         4
persistent-superblock 1
device             /dev/app01vg/test2
raid-disk          0
device             /dev/nd10
raid-disk          1

raiddev /dev/md1
raid-level          1
nr-raid-disks      2
nr-spare-disks     0
chunk-size         4
persistent-superblock 1
device             /dev/app01vg/clcst
raid-disk          0
device             /dev/nd1
raid-disk          1
```

1 Cluster Resource Management


26.5 Application IP Address Management

It should first be noted that the cards defined for the cluster can be used for more than one application simultaneously. This is handled by using aliasing on the card.

This is probably most of the most crucial areas of managing an application instance. This software has been designed around offering the following two mechanisms for handling failure of a card which is currently being used for one or more IP addresses.

There are two different methods that will be supported by the software - the first is recommended as it offers greater resilience since much of the work is handled by the kernel. The second offers similar functionality but occurs from user space, meaning that it works, but tends to be much slower. However the advantage of this approach is that it can be modified by the user to work in different ways. The two mechanisms for IP address management are:

- Channel Bonding - use of the kernel “bonding” driver to join cards together to make a single logical card.
- IP Address take-over - by periodically checking the link status of the active card a fail-over to an alternative card can be performed.

 As of 0.7.0 channel bonding for application IP address handling is not fully implemented and thus is not yet available.

26.6 NBD / ENBD Server Management

Whenever a system is a standby for the current live environment it must run (e)NBD servers for every file system that belongs to the application. To make this management there is a utility called “svrstart” which can be used to start these (e)NBD server - or restart them.

To function correctly it makes use of a local directory called:

```
/cluster/control/.resources/pids
```

In this directory each server will have a separate file as follows:

```
Application.volumegroup.volume.nbd
```

The contents of the file is the process ID of what the process that is (or was last) running that server. If the specified file does not exist then this will indicate that the service has not been run, or has been shutdown cleanly.

On the machine currently serving the application the relevant “client-side” processes will be running. These will also use entries in the above directory, but the format of the names will be slightly different - the extension will be “cnbd” rather than “nbd” to denote “client”, eg:

```
Application.volumegroup.volume.cnbd
```

27 Kernel / System Software Configuration

The intention of this section is to describe the configuration of the kernel that is necessary to ensure that the clustering software will function on a particular pair of machines. The main point to remember is that no particular distributed is recommended - just that the kernel used is configured in the manner specified below.

27.1 General Configuration


Since the software recommends use of the TCP version of the “Ping” function, you need to ensure that the “echo” request is made available. This is typically found in the “/etc/inetd.conf” file, or the “/etc/xinetd.d/echo” file if using “Xinetd” rather than “inetd”.

For “inetd” the entry must appear as follows:

```
echo stream tcp nowait root internal
```

For “xinetd” the file contents should be as follows:

```
service echo
{
    disable          = no
    type             = INTERNAL
    id               = echo-stream
    socket_type     = stream
    protocol        = tcp
    user            = root
    wait            = no
}
```

 You may need to refresh the daemon or restart the system to ensure these changes are reflected by the “inetd” super daemon in use.

More information on use of the relevant use of the “ECHO” service can be found during the installation instructions, starting on page .

27.2 The “Raid” and “LVM” Modules

If it recommended that the following modules be installed as part of the boot process on Linux 2.4-based environments:

- md
- raid1 / fr1

On Linux 2.6-based servers the following modules are required as a minimum for “linuxha” functionality:

- dm_mod
- raid1 / fr1
- md

The “enbd” module is installed into the kernel as part of the application started on Linux 2.6-based servers.

In either 2.4 or 2.6 Linux versions it does not matter whether LVM version 1 or version 2 is used, nor whether the functionality is compiled in or loaded in directly as a module. However, if module based functionality is chosen it should be loaded as part of the standard system start-up process.

27.3 The “bonding” Module

Firstly the kernel must be compiled with the “bonding.o” module present. This can be found in the following local:

Network Device Support -> Bonding driver support

Once available the “clbuildapp” utility when bonding is mentioned as an option for IP availability will check and configure the local / remote machines with information such as the following in /etc/conf.modules:

```
alias bond0 bonding
alias bond1 bonding
alias bond2 bonding
```

Given that a typical machine will not have more than 6 network cards, three bonded devices is usually more than enough!

Whenever the software activates a bonding device it will provide all necessary kernel module options on the command line.

28 Custom Perl Modules

28.1 The "fsmap" Module

This module is used for resource allocation queries to ascertain exactly what is allocated to a particular application in the cluster. As of version 0.6.3 of the "linuxha" toolset it will also allow querying of remote resources as well.

The module creates an object via the standard "new" method:


Name:	new
Purpose:	Returns a blessed reference of type "fsmap".
Inputs:	Hash list consisting of the following: FILE – The name of the file to read in containing the "fsmap" for the application. This is typically the file "/cluster/control/.resources/fsmap/<application>". APPLICATION – The name of the application. HOST [Optional] – Allows the object to scan the contents of the specified node, rather than the local node.
Outputs:	Blessed reference, or the "undef" value if an error has occurred.

Once created the object understands the following methods:

Name:	get
Purpose:	Returns an a list of details for the specified record: VG – The volume group for the specified file system. LV – The logical volume name of the specified file system. MNT – The mount point for this file system. TYPE – The file system type, such as "reiserfs" or "jfs" MD – The RAID device number allocated to this file system on this host. ND – The ENBD device allocated to this file system. These numbers start at 0 - the ENBD device name can be mapped to a device name via the "mapdevicename" function offered by the "clutils" module. OPTS – The file system mount options, if any. PORT – The network port allocated on this port when running the ENBD service for this file system.
Inputs:	Hash list consisting of the following: RECNO – The record number, (first is 0), of the details to return.
Outputs:	If the record specified exists then the list of details as specified above is returned. If no such record exists then the "undef" value is returned.

Name:	get_md_list
Purpose:	Returns a list of RAID devices for the application.
Inputs:	None.
Outputs:	A list of RAID device numbers allocated to this particular application currently.

Name:	get_md_entry
Purpose:	Returns all details for a particular RAID device, given the RAID device number.
Inputs:	Hash consisting of just one mandatory element:

Name:	get_md_entry
Outputs:	MD – The device number of the RAID device to return details of. If the specified device exist, then a list of the following elements, (as described for the “get” method previously). VG, LV, MNT, TYPE, MD, ND, OPTS, PORT
Name:	reread
Purpose:	Rescans the resources allocated to the particular application.  Currently this is only supported when the fsmap details are being queried for the local host.
Inputs:	None.
Outputs:	Undef – An error occurred whilst scanning. 0 – Reread information successfully.
Name:	count
Purpose:	Returns a count of the number of file system allocated for this application.
Inputs:	None.
Outputs:	Count – integer 0 or above.
Name:	nbd_belongs
Purpose:	Returns the RAID device number when given the ENBD device number.
Inputs:	NAME – The device name of the nbd device, such as “a”, or “ba”.
Outputs:	The RAID device number for this ENBD device, or -1 if the specified ENBD device entry can not be found.

Name:	<code>get_fs_match</code>
Purpose:	Returns the record number of the information for the specified file system, given the mount point.
Inputs:	Hash with the following single mandatory argument: <code>FILESYSTEM</code> – Mount point to return details of.
Outputs:	<code>>=0</code> – the record number of the matched entry. <code>-1</code> – indication that the file system is not recorded for the application.

28.2 The “clutils” Module

This module contains a large number of routines that are used by many of the main scripts used by the “linuxha” application.

Name:	
Purpose:	
Inputs:	
Outputs:	

28.3 The “clbonding” Module

This module contains a series of routines that are used by various cluster tools to check, create, remove and monitor the state of a bonding configuration. This is a standard module but all routines must be imported explicitly.

Name:	<code>bonding_check_params</code>
Purpose:	Checks to see if the list of values, (see format below), are valid to be used to be passed to form a new bonding device.
Inputs:	<code>String</code> - a string containing the “param:value” settings, white space separated.
Outputs:	An array containing either: <code>(0, "OK")</code> - the parameters passed are acceptable, or; <code>(1, "err", [...])</code> - one or more parameters are not recognised or have invalid values.

Name:	<code>bonding_get_defaults</code>
Purpose:	Returns a string of the defaults to use for bonding parameters (such as the “mode”) if the application does not define any.
Inputs:	None.
Outputs:	<code>String</code> - containing the default values - as understood by the <code>bonding_check_params</code> routine.

Name:	<code>bonding_create</code>
Purpose:	This will create a new device and return the name of it - such as “bond2”. This is the network interface that has been created, (or re-created if it already existed) that has been assigned the IP address for the package. The checks include checking to ensure that no routes are currently defined against the cards, (which insures they are not currently in use).
Inputs:	Hash with the following mandatory elements: <code>IP</code> - The IP address to assign to the new interface. <code>APPLICATION</code> - The name of the application, (which is used to load the bonding module with a unique name).

Name:	<code>bonding_create</code> INTERFACES - A comma-separated list of exactly two interfaces to use as the bonding network configuration. FLAGS - A series of white space separated "attribute:value" pairs which define the type of bonding interface to create. It is assumed they these have already been validated via the "bonding_check_params" function. The following is optional: NETMASK - The Netmask to apply to the new interface that has been created.
Outputs:	If an interface has been created successfully then the return list will be: ("OK", "interfacename") If an error has occurred, then the following is returned instead: ("ERROR", "Error message")

Name:	<code>bonding_get_interfaces</code>
Purpose:	Returns a list of two interfaces that are not currently in use - this is defined as them having no entries in the routing table. These are then used as candidates to bonding device creation (probably).
Inputs:	String - a comma-separated list of interface names to check.
Outputs:	([eth0], [eth1]) - list containing two available interfaces - empty if none are available, or 1 element long is only one is available.

Name:	<code>bonding_destroy</code>
Purpose:	Removes the specified bonded interface, given just the IP address hosted and the application name using it. It will also ensure the module in question is unloaded (if possible), and ensure that any interfaces slaved have their IP address set to 0.0.0.0 allowing them to be re-used for other bonded network interfaces when necessary.
Inputs:	Hash with two mandatory arguments: <code>APPLICATION</code> - The name of the application with the bonded interface. <code>IP</code> - The IP address associated with the bonded interface.
Outputs:	(<code>"ERROR"</code> , <code>"message"</code>) - unable to removed the bonded interface - <code>"message"</code> is text explanation why. (<code>"OK"</code> , <code>"interface"</code>) - the bonded interface <code>"interface"</code> has been removed successfully.

Name:	<code>bonding_get_info</code>
Purpose:	Returns information about the specified bonded device as a list.
Inputs:	Hash with one mandatory arguments: <code>INTERFACE</code> - The name of the bonded interface - such as <code>"bond0"</code> .
Outputs:	(<code>"ERROR"</code> , <code>"message"</code>) - unable to find details of the specified bonded interface (<code>"mode"</code> , <code>"active"</code> , <code>"int1"</code> , <code>"f1"</code> , <code>"int2"</code> , <code>"f2"</code>) where; <code>mode</code> - text mode of bonded interface. <code>active</code> - the currently active interface (or <code>"both"</code>). <code>int1</code> - the ethernet device of the first interface. <code>f1</code> - the link failure count of the first device. <code>int2</code> - the ethernet device of the second interface. <code>f2</code> - the link failure count of the second device.

29 Application Directories

This is a short section that describes the extensions used by the software, the directory structure of the installed code, and any default log files that are generated, and how they should be managed. It should be noted that this product is still under heavy development and this information is subject to change for later versions. This document covers release 0.5.0 of the "linuxha" products, (and the associated "linuxhatools") only.

29.1 Non-Standard Perl Packages

The table below lists the required Perl packages that must be installed before the "linuxha" software can be used. All these packages are made available as part of the "linuxhatools" package, installation of which is covered on page .

Name	Author(s)	Description
XML-Parser-2.31		Provides XML::Parser module for handling XML files (which are used for all configuration information)
expat-1.95.6		Provides the low level C library used for the XML-Parser module.
Time-HiRes-1.43		Provides the Time::HiRes module which is used for scheduling purposes when the current time to fractions of a second is required.
Crypt-CBC-2.08		The package used to handle the encryption of variable length data into blocks.
Net-ext-1.011		The collection of modules to make Socket-based servers easier to author.
Crypt-Blowfish-2.09		The Blowfish encryption algorithm used for the encryption.

Directories used by the product

Directory	Contents
<code>/etc/cluster</code>	Configuration files for packages and the definition of the cluster.
<code>/etc/cluster/<package></code>	Per application configuration files for the cluster.
<code>/etc/cluster/<package>/flags</code>	Directory used for flag files for the "Lems" flag_check module.
<code>/usr/local/cluster</code>	This directory and any subsequent sub directories are used to contain perl modules that are not "installed" into the standard Perl installation - XML::Simple.pm for example.
<code>/usr/local/cluster/lib</code>	
<code>/usr/local/cluster/lib/perl</code>	
<code>/sbin/cluster</code>	Directory containing standard binaries to use with the clustering software.
<code>/sbin/cluster/utlils</code>	Utility scripts some of which can be used on the command line, but typically utilised by the programs in the parent directory.
<code>/usr/src/cluster</code>	This directory contains several sub-directories that are installed when the prerequisite "linuxhatools" package is installed, (each exemplified below).
<code>/usr/src/cluster/raid</code>	Patched version of the "raidtools" utility set used by linuxha. This directory is used to compile and install and is left as a reference of the source after the installation is complete.
<code>/usr/src/cluster/perl</code>	Contains copies of the Perl modules that are installed for "linuxhatools" – as listed previously.
<code>/usr/src/cluster/nbd</code>	Contains patched version of the ENBD product that is required for "linuxha". Again these tools are compiled and installed when the "linuxhatools" package is installed.
<code>/usr/src/cluster/lvm2</code>	Patched version of the LVM2 toolset that linuxha uses. Only difference is that device scanning omits RAID and ENBD devices, since this is known to cause deadlocks on occasion.

<code>/usr/src/cluster/doc</code>	Basic README and INSTALL instructions, will contain other documents in later releases.
<code>/usr/src/cluster/expat</code>	The "Expat" tool-set that is used for the XML::Simple parser module that "linuxha" uses.

Perl Modules Used

Module	Purpose
Digest::MD5	Used to checksum configuration files to ensure they are the same on both nodes in the cluster.

Non-Standard Unix Commands Required

Name	Purpose
Md5sum	Return the MD5 checksum of a local file.
Vgcreate	Create an LVM Volume Group
Nbd-client	Create a nbd Client connection
Nbd-server	Serve a local block device as an nbd resource
Enbd-client	
Enbd-server	
Raidstop	
Raidstart	
Raidsetfaulty	
Raidhotadd	

A. Understanding “ENBD”

I. Introduction

The purpose of this appendix is to describe how I believe “enbd” works, and the changes introduced against the standard features to make it more compatible with the goals and functionality of the high availability software.

This section is based entirely on my own understanding of “enbd” and thus is likely to be subject to change as I learn more about how “enbd” actually works!

II. An “Enbd” client / server Example Walkthrough

Like “nbd”, “enbd” obviously requires that the server is started before the client on a port specified on the command line. Although the server listens on the specified port, this is not the whole story, as the scenario below demonstrates.

Typically to start a server process the following command line would be used:

```
# enbd-server -P 9921 -E /dev/app01vg/test2 -b 1024 -F 9950 -i hello
```

Note: This command line is different than the standard enbd-2.4.30 software would parse, as explained in a later sub-section.

This command will indicate that the block device “/dev/app01vg/test2” will be served by “enbd” on port “9921”. The “-b” indicates the block size, 1024 being suitable for standard Ethernet networks, (i.e. smaller than the MTU to prevent loss of performance due to fragmentation).

Finally the “-F” indicates that the software should examine ports from 9950 upwards when looking for ports to dynamically allocate to the client connection requests. This is followed by the “-i” flag which indicates an identity to be associated with this request to ensure only clients with a matching identity can connect.

On running this command line the following will be shown on the terminal:

```
enbd-server 218: server (-2) locked /var/state/nbd/server-
hello.client_ips
enbd-server 218: server (-2) pinged service nbd-cstatd at
172.16.235.101:5051
enbd-server 218: with news "notice server-start 9921 127.0.0.1
172.16.235.100 172.16.235.110
quit
"
enbd-server 218: server (-2) unlocked /var/state/nbd/server-
hello.client_ips
enbd-server 218: server (-2) set new signal handlers for master
server 218
```

Once this command has been run the specified process is now running in the background. A check with netstat indicates it is listening on the expected port:

```
# netstat -an|grep 9921
tcp        0      0 0.0.0.0:9921          0.0.0.0:*            LISTEN
```

At this point the client is able to attempt a connection. To do this the following command line is used on the client machine:

```
# enbd-client -S servera -P 9921 -n 4 -i hello -D /dev/ndb
```

In this instance the server to connect to is "servera", and the port the "enbd" service is listening on is "9921". The "-n" is the number of channels to use - 4 is the typical recommendation. Finally the "-i" option is used to identify the client with the actually "enbd" device given on the end of the command line.

As soon as this command is entered output will appear on both the client and the server as they attempt to negotiate the characteristics of the connection. The following characteristics are considered:

- *The recommended block size* - The value chosen will be the larger of the client / server values. If the "-b" option is not specified when starting the client or the server it defaults to 1024.
- *The pulse interval* - how often the client attempts to contact the client to see if it is still active. If the server is unable to contact the client I believe the client will go dormant waiting for a connection to respond?
- *The number of channels* - The number of channels indicates how many processes are used on both the client and server to service this connection. Four are typically used since this improves concurrency when dealing with multiple requests.

Once the connection has been established the device file specified on the command line on the client can be used as expected, i.e.:

```
mount /dev/ndb /tmpmnt
```

III. Stopping the Client

To stop the client, then the following series of actions are recommended.

The first step is to stop using the device. This is achieved by un-mounting the file system:

```
umount /tmpmnt2
```

It should be noted that un-mounting the file system might take a few seconds, since the flushing of buffers requires all communication to the other server to be validated and completed.

Once the file system has been unmounted, issue an additional "sync" command just in case:

```
sync
```

Finally to stop the client issue the following command:

```
kill PID
```

The process ID to choose is the one that is serving the resource and has several children - this is the process that has a Parent Process ID of 1. Several seconds after killing it the client and server resources will be de-allocated on both sides for this connection.

Of course this will still leave the server process ready to serve another connection to this resource. Stopping the server resource will cause all client resources to be terminated as well.

IV. Understanding Device Resource Allocation

One of the most important points to understand when using "enbd" is that each server/client connection must use multiple "/dev/nd" devices on the client. The configuration used as part of this software is configured for the kernel module, client and server to support up to 64 devices - each having four paths for availability.

Thus when you connect a client using the device "/dev/nda", actually the following four devices will actually be used, (each with a separate client/server process):

```
/dev/nda  
/dev/nda1  
/dev/nda2  
/dev/nda3
```

When more than 26 devices are available for format is as follows:

Devices 1-26 are “/dev/nd[a-z]”, whilst devices “27-52” are “/dev/ndb[a-z]”. The next set of devices, 53-78” are “/dev/ndc[a-z]”. Thus when using the configuration supplied here the devices available range from “/dev/nda” through to “/dev/ndcl”, (1 through 64).

Unfortunately the standard “MAKEDEV” script supplied with “enbd” is not capable of generating the required block devices for this configuration. To alleviate this problem the patched version includes a “MAKEDEV.pl” which is able to generate device files necessary to this configuration.

To generate all necessary device files, change to the directory where the patched version of the software is installed, (i.e. “/usr/local/src/enbd-2.4.30-patched”), and run the following command to create all the devices:

```
./MAKEDEV.pl --device 0 --range 64 -v
```

Note: *The “range” argument can be removed to create a single device. The device numbers go from “0” through to 63.*

If any of the the files already exist an error will be given and the program abort.

B. Using Raid Modules for Data Replication

I. Introduction

The purpose of this short appendix is to describe in detail the use of the current raid modules under Linux 2.4 to support the data replication, which is used in this software along with "enbd" or "nbd" as mentioned to replicate changes to a backup machine.

RAID functionality has been available for Linux for several years - even so there is still room for improvement, as explained in detail throughout this section.

II. Basic Raid Operations

Before describing the actions used by the software, the basic available actions for a RAID array will be explained.

III. Understanding /proc/mdstat

Use of this status file is critical for the application software, since it determines the status of every raid device currently in use on the system. When all raid devices are operating normally the contents will look similar to the following:

```
Personalities : [raid1]
read_ahead 1024 sectors
md10 : active raid1 lvmb[0] ndk[1]
      53184 blocks [2/2] [UU]

md0 : active raid1 lvma[0] nda[1]
      20416 blocks [2/2] [UU]

unused devices: <none>
```

In the above example there are two RAID devices currently in use on the server, both of which are functioning normally. The first two lines describe the available "personalities" - either "raid1" or "fr1" **must** be included here for the clustering software to function.

After the first two lines each device is then allocated two or more lines. Following the device name the status of the device is listed, which is typically "active". Following is the personality of this device, which will be "raid1" or "fr1" for clustered devices, the actual device names, (without the "/dev" components are shown).

The actual names of the devices are simply referred to as "lvma" for logical volumes. The numbers in square brackets indicate the device number in the array.

The second line indicates the size of the array, (each block is 1Kb), following by the devices in the array and usable devices, (always "[2/2]" when things are synchronised). The final set of square brackets use a single character for each device, where "U" indicates up to date, whilst "_" means not synchronised.

The following example shows the contents of the file when one of the RAID1 devices is undergoing synchronisation:

```
Personalities : [raid1]
read_ahead 1024 sectors
md10 : active raid1 lvmb[2] ndk[1]
      53184 blocks [2/1] [_U]
      [=====>.....]  recovery = 76.9% (41688/53184) finish=0.3min speed=592K/sec
md0  : active raid1 lvma[0] nda[1]
      20416 blocks [2/2] [UU]
```

When a device is being resynchronised the contents of `/proc/mdstat` show how are through the resynchronisation the software is, both as a percentage and as a number of blocks synchronized, compared to the total block count.

Interestingly the average rate of synchronisation and the estimated completion time are also recorded. The estimated completion time is accurate to 1/10th of a minute, (6 seconds).

IV. Synchronising Multiple Devices

Given that most applications will consist of more than a single file system, there may be occasions when more than one RAID device needs to be synchronised. This is the case, for example, when the application is being built, (via `clbuildapp --sync`), or if a secondary node for the application comes back into the cluster.

In such cases the `raidhotadd` command will be run against several arrays simultaneously. In such situations you will notice that the `/proc/mdstat` may look similar to the following:

```
Personalities : [raid1]
read_ahead 1024 sectors
md1  : active raid1 lvmc[2] ndb[1]
      8128 blocks [2/1] [_U]

md10 : active raid1 lvmb[2] ndk[1]
      53184 blocks [2/1] [_U]

md0  : active raid1 lvma[2] nda[1]
      20416 blocks [2/1] [_U]
      [=====>.]  recovery = 95.0% (20416/20416) finish=0.0min speed=2268K/sec
unused devices: <none>
```

The key point here is that despite all three devices being active with faulty mirrors the resynchronisation code, (handled via the `raid1d` process will only synchronise one device at a time - all other devices will be handled once the current one is complete.

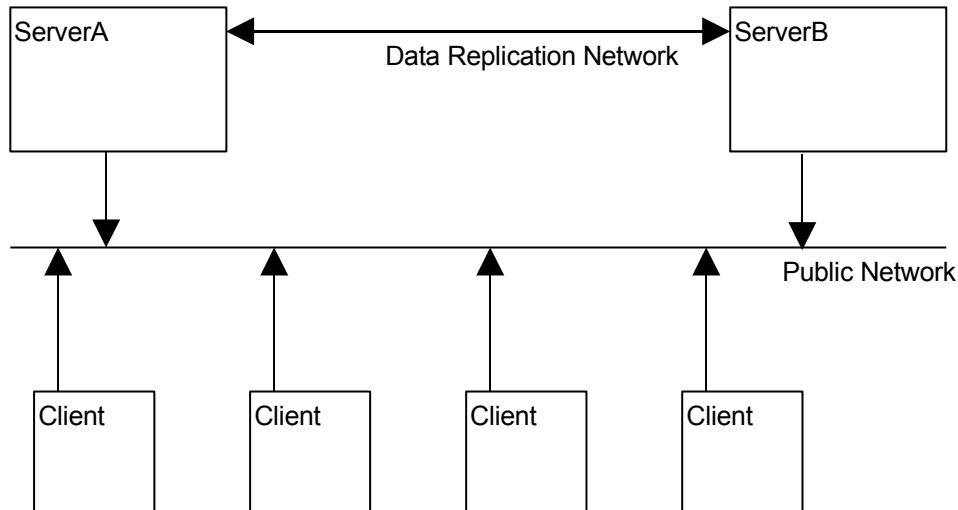
Thus the ordering of synchronisation is based on the order of the `raidhotadd` command, which in turn, is based on the contents of the `fsmap` for the application. This in turn is based on the order the file systems are mounted when the application is built. This fact should be borne in mind when building the application and leads to the following points:

Hint: *if you wish to secure certain files systems as soon as possible, mount them as soon as possible.*

C. Setting up SSH

I. Introduction

One of the requirements of this clustering software is to ensure that it is possible for both servers to communicate to one another over all communication channels. Usually there will be at least two such channels, as the diagram below demonstrates:



Notice that in the above (common) example, the hosts typically communicate using the replication network, but they can also communicate via the other interfaces/IP addresses defined for the application/topology.

The steps below allow both servers to use the “ssh” and “scp” commands between one another as “root” without prompting for a password or pass phrase. This is absolutely essential for the clustering software to work - and if these or similar steps (if you have a preferred alternative method for ssh configuration), are not following the cluster software will not build correctly.

All steps below should be run as “root” on each machine in the cluster.

II. Step 1: Define Public/Private Key Pairs

The first step is to define a public/private key pair for each machine in the cluster. This is as simple as running the following command on each server in turn:

```
# ssh-keygen -t rsa -b 1024
```

When run you will then see the following output (or something very similar to the following):

```
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
```

In this instance simply hit <RETURN> to accept the default file location. This will be followed by the following output:

```
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
```

Again simply hit <RETURN> and then again when asked to repeat the key phrase. You will then be shown the public portion of the key that has been generated.

Now repeat this step on the other node in the cluster before continuing.

III. Defining Machine Definitions

To ensure that this host is known to both itself and the other node in the cluster each machine in turn must be connected to. To attach to the local machine, run the following command:

```
# ssh localhost ls
```

If this is the first time this has been tried output similar to the following will appear:

```
The authenticity of host 'localhost (127.0.0.1)' can't be established.
RSA key fingerprint is 53:86:82:ea:80:a5:cd:26:14:5e:46:07:b4:54:8c:10.
Are you sure you want to continue connecting (yes/no)?
```

Enter “yes” if this prompt and then enter the current local “root” password to finally run the command.

Now we need to perform the same steps for the other node in the cluster, for example “serverb”:

```
# ssh serverb ls
```

Again enter “yes” and then the current “root” password for that server.

Now repeat this step on the other node in the cluster, obviously using the name of the first node when running the final “ssh” command just given.

IV. Define SSH client authorization

The final step is to ensure that each host allows one another (and itself again!) to use ssh without requiring a password. This is achieved by copying the public key part of the public/private key pair to login by providing the correct pass phrase defined for this particular host.

Since the pass phrase entered above was empty this will in affect allow us to log in without stopping to provide input - absolutely required for running the software in the cluster environment!

To do this list the public key of the local machine, run the following commands:

```
# cd ~/.ssh
# cp id_rsa.pub authorized_keys
```

Now this file should be edited to include the contents of the “id_rsa.pub” key copied from the other node. Once completed the file will look similar to the following file, containing just two long lines of output:

```
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEAlj9iflPPIF4Y9ALe9QHfSQiX011I44RPbLP/oor4WRguyv/Rwp
EQE3aBsvQy551NLDVoBXGx54UBD/HLtW8lckKUefwQm4txqTrNRFzTyDzWr71MXKx/7B04ro0Nt3SPz2WTSeJC
Etvv5DC0LxkJXnq1BZRoeGxBgiLKYGndsbc= root@servera
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEAUecNjrcOzEabCZnVAoOef4/+sqZpEHnv2Igr0GJYeVxI911CJN
lwjgGbEahz9vSHSYBlGtRHXmz/ohVBqCPfP2TKiQieR+S2YncwkdDOR5V6Q6B8ILKuzJSD24pLfo7aZpg2pzQ4
YH5TbWwAtbs6tqaldJc+cVySZi2Ty8R0xRs= root@serverb
```

This step should now be repeated on the other node to ensure “ssh” works in a similar way for that host as well.


This should complete the basic set up to allow both hosts to use the “ssh” and associated programs without requiring a pass phrase or password to be entered each time.

If you need to allow other servers to access these hosts in a similar way simply add their public keys for the user in question to the "authorized_keys" file on each host as shown above.

At this point it is worth checking the configuration attempting to "ssh" to all configured addresses. For example, simply use the following command again from "servera" to "serverb":

```
servera# ssh serverb ls
```

Please remember that if multiple IP addresses are available on each host the steps should be repeated for all addresses on all hosts. Although time consuming it only has to be done once.

 If the above steps do not work double check the "authorized_keys" file on each host – a common mistake is to cut and paste the lines – this might introduce end-of-line characters and each entry needs to be on a single line.

D. Guidelines for Editing XML Files

The purpose of this short appendix is to give some straightforward rules that should be followed when editing the XML files used for the "linuxha" product. This information is only useful to administrators who are not currently aware of the syntax rules that govern the use of XML files.

<more on characters, comments, etc>

E. Sample “Apache” Application Configuration

F. Introduction

The purpose of this section is to describe the process of creating the environment ready to use the “clbuildapp” routines to create a package that will be used to start a sample “apache” application in the cluster.

G. Building the Volume Group

The steps here assume that you are using “LVM version 2” (device mapper). If you are using LVM version 1 instead the commands used will be slightly different. The differences between the two command sets will be shown.

Note: *All commands here should be carried out on the first host in the cluster - any commands that need to be run on the other host will be specifically mentioned.*

The first step is checking to see if you have a spare “physical” volume which you can assign to the volume manager. On the sample server we have configured the “sda” drive, (the first SCSI device) as follows:

```
# fdisk -l /dev/sda

Disk /dev/sda: 4194 MB, 4194892800 bytes
255 heads, 63 sectors/track, 510 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1            1         262     2104483+   83  Linux
/dev/sda2           263         279     136552+    82  Linux swap
/dev/sda3           280         510    1855507+    5  Extended
/dev/sda5           280         305     208813+    8e  Linux LVM
/dev/sda6           306         331     208813+    8e  Linux LVM
```

If in this case the first drive is used to boot the system, but also provides two partitions “sda5” and “sda6” which can be used as “physical” devices in LVM. You can use complete drives though if you have them spare, but this is not strictly necessary.

We wish to create a volume group called “app01vg”, and to do this we must first indicate that “sda5” and “sda6” are to be used as “physical devices” by the volume manager. This is done using the following command:

```
# pvcreate /dev/sda5
No physical volume label read from /dev/sda5
Physical volume "/dev/sda5" successfully created
# pvcreate /dev/sda6
No physical volume label read from /dev/sda6
Physical volume "/dev/sda6" successfully created
```

Once the devices have been created the volume group can be created. Here we just use a single "physical volume" - though of course others can be added now, or at a later date:

```
# vgcreate app01vg /dev/sda5
Volume group "app01vg" successfully created
```

Now the "logical volumes" (one per file system) can be created. This is just as straightforward. We give them names of "admin", "logs" and "docs" in this instance:

```
# lvcreate -n admin -L 20 app01vg
```

If this is the first time you have used LVM and are using LVM version 2, you might get the following error:

```
/dev/mapper/control: open failed: No such file or directory
Is device-mapper driver missing from kernel?
Failed to activate new LV.
```

This occurs because there is an entry missing from /dev. To ensure that this is created the "device-mapper" distribution comes with a file called "devmap_mkknod.sh" under the scripts directory. This must be run to create the required "control" device on each server in the cluster:

```
# /tmp/device-mapper.1.00.07/scripts/devmap_mkknod.sh
```

Of course you will need to change the path to that relevant to where you unpacked the device mapper if you installed it manually. You will not need to re-run the previous command since it created the logical volume, just not activated it. We carry on to create the other logical volumes, each also 20Mb in size:

```
# lvcreate -n docs -L 20 app01vg
Logical volume "docs" created
# lvcreate -n logs -L 20 app01vg
Logical volume "logs" created
```

If you did get the activation error above then we need to active the logical volume manually:

```
# lvchange -a y /dev/app01vg/admin
```

All the above commands were for LVM v2 - LVM v1 commands are the same apart from the "lvcreate" commands - here we need to replace the "app01vg" at the end of the line with "/dev/app01vg" in each case. For example:

```
# lvcreate -n admin -L 20 /dev/app01vg
```

Now the logical volumes have been created we can generate file systems on them. To the operating system each logical volume appears to be like a complete disk, and so we use the standard commands to create the file system, just specifying the name of the logical volume instead of the name of a disk partition. Due to the size of the logical volumes, (20Mb each), we have chosen to use "Jfs" as the file system type since Reiserfs does not appear to work well on such small file systems:

```
# mkfs -t jfs /dev/app01vg/admin
# mkfs -t jfs /dev/app01vg/docs
# mkfs -t jfs /dev/app01vg/logs
```

You will need to answer "Y" for each command to confirm the action. If these commands fail you will need to download the JFS file system and user space tools from the IBM web site:

<http://www-124.ibm.com/developerworks/oss/jfs/>

This website includes information on how to install the file system and user space tools.

Once the file systems have been created to create the mount points we expect them to use:

```
mkdir -p /apache /apache/docs /apache/admin /apache/logs
```

Now we finally mount these file systems on the first machine ready to install the software that will be clustered:

```
# mount /dev/app01vg/admin /apache/admin/
# mount /dev/app01vg/docs /apache/docs
# mount /dev/app01vg/logs /apache/logs
```

Now the file systems are mounted and ready to use. If you are using LVM v2 you will notice that the devices you used to mount the file systems in the above commands are not the devices shown via "df" - this is expected:

Filesystem	1k-blocks	Used	Available	Use%	Mounted on
/dev/sda1	2104408	1245564	858844	60%	/
/dev/mapper/app01vg-admin	19248	136	19112	1%	/apache/admin
/dev/mapper/app01vg-docs	19248	136	19112	1%	/apache/docs
/dev/mapper/app01vg-logs	19248	136	19112	1%	/apache/logs

Finally we must actually create the volume group on the other node - so run the following commands on the other node in the cluster:

```
# pvcreate /dev/sda5
# vgcreate app01vg /dev/sda5
```

No other commands are necessary - the process of building the application described starting on page will create the logical volumes and mount points on the other node if necessary.

For more information on the flags that can be used with the "lvcreate", "pvcreate" and "vgcreate" see the appropriate manual pages.

H. Using the Sample files

Once you have mounted the file systems you will be able to copy the sample files that come with the "linuxha" package over to a suitable directory. Use the following commands to create a configuration directory on the clustered file system, (which was mounted on the first node in the previous section).

We also take this opportunity to create a directory for the administration scripts:

```
# mkdirr /apache/admin/conf /apache/admin/scripts
# chmod 755 /apache/admin/conf /apache/admin/scripts
```

Now the configuration files can be copied using the following command:

```
# cp /etc/cluster/apache/conf/* /apache/admin/conf
# chmod 444 /apache/admin/conf/*
```

You will need to change the "httpd.conf" to provide details of the correct locations of where files, (such as the home page and the logs) will be stored. Below are the two lines that I changed to get my example working:

```
httpd.conf:ServerRoot "/apache"
httpd.conf:DocumentRoot "/apache/docs/htdocs"

httpd.conf:<Directory "/apache/docs/htdocs"> (for document root)

<IfModule mod_mime.c>
    TypesConfig /apache/admin/conf/mime.types
</IfModule>

<IfModule mod_mime_magic.c>
    MIMEMagicFile /apache/admin/conf/magic
</IfModule>

ErrorLog /apache/logs/error_log
```

```
CustomLog /apache/logs/access_log common

# mod_alias.c settings...
<IfModule mod_alias.c>
  Alias /icons/ "/apache/docs/icons/"

  <Directory "/apache/docs/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
  </Directory>

  ScriptAlias /cgi-bin/ "/apache/docs/cgi-bin/"
  <Directory "/apache/docs/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
  </Directory>
</IfModule>
```

Please note that the changes that were made enabled a simple web server session to be validated in the cluster. It is likely that a full production system would need to further changes to the configuration files in use.