

## High Availability Solutions with SteelEye's LifeKeeper for Linux

### Index

<i>Introduction</i>	<i>Page 2</i>
<i>LifeKeeper for Linux</i>	<i>Page 4</i>
<i>Overview</i>	<i>Page 4</i>
<i>LifeKeeper History</i>	<i>Page 4</i>
<i>Lifekeeper Core Software</i>	<i>Page 5</i>
<i>Application Recovery Kit Software (ARKs)</i>	<i>Page 5</i>
<i>Bespoke Applications</i>	<i>Page 7</i>
<i>Clustering Technology</i>	<i>Page 8</i>
<i>What clustering achieves</i>	<i>Page 8</i>
<i>How clustering works</i>	<i>Page 8</i>
<i>Cluster Configurations</i>	<i>Page 8</i>
<i>N+1</i>	<i>Page 8</i>
<i>Active/Standby</i>	<i>Page 9</i>
<i>Active/Active</i>	<i>Page 9</i>
<i>Disaster Recovery Solution</i>	<i>Page 9</i>
<i>Virtual Servers</i>	<i>Page 11</i>
<i>Service Virtualisation</i>	<i>Page 11</i>
<i>Significance of Virtual/Floating IP addresses</i>	<i>Page 12</i>
<i>I/O Considerations</i>	<i>Page 12</i>
<i>Shared Storage</i>	<i>Page 12</i>
<i>Data Replication</i>	<i>Page 13</i>
<i>Synchronous Replication</i>	<i>Page 13</i>
<i>Asynchronous Replication</i>	<i>Page 13</i>
<i>Bandwidth Requirements for Data Mirroring</i>	<i>Page 14</i>
<i>I/O Fencing</i>	<i>Page 14</i>

## Index continued

<i>Clustering Operation</i>	<i>Page 15</i>
<i>Overview</i>	<i>Page 15</i>
<i>Load Balancing</i>	<i>Page 16</i>
<i>IP Availability</i>	<i>Page 16</i>
<i>Resource Hierarchies</i>	<i>Page 16</i>
<i>Failover Mechanics</i>	<i>Page 17</i>
<i>Intra cluster Communication</i>	<i>Page 18</i>
<i>Resource Checking</i>	<i>Page 18</i>
<i>Fail over Times / Performance</i>	<i>Page 19</i>
<i>Distributed Cluster Meta Data Storage Model</i>	<i>Page 19</i>
<i>Split Brain Protection</i>	<i>Page 19</i>
<i>Active / Passive Split Brain with Shared Storage</i>	<i>Page 20</i>
<i>Active / Active Split Brain with Shared Storage</i>	<i>Page 20</i>
<i>Active / Passive Split Brain with Data Replication</i>	<i>Page 20</i>
<i>Active / Active Split Brain with Data Replication</i>	<i>Page 20</i>
<i>Management</i>	<i>Page 21</i>
<i>Acknowledgements</i>	<i>Page 25</i>
<i>About Open Minds High Availability Solutions</i>	<i>Page 26</i>
<i>About SteelEye Technology Inc.</i>	<i>Page 26</i>
<i>About this document</i>	<i>Page 26</i>

## Abstract

Clustering as a technology has been available in one form or another for probably the last 15-20 years, but the explosive growth of computing in every day life it has recently seen it's usage flourish. This paper gives an overview of High Availability Clustering for Linux and contains a technical overview of SteelEye Technology Inc's LifeKeeper for Windows and discusses hardware and software configurations and general clustering technology.

## Audience

Readers are expected to be from the IT field, and have a general understanding of computing environments.

## Introduction

This document aims to give you, the reader, an overview of High Availability Clustering for Linux covering some of the issues and concepts involved.

Clustering as a technology has been available in one form or another for probably the last 15-20 years, but the explosive growth of computing in every day life it has recently seen it's usage flourish. There are two basic forms of clustering - either the processing power of multiple computers is harnessed together (High Performance Clustering), or to provide redundancy and fault resilience (High Availability Clustering). For the purpose of this paper we are only concerned with High Availability Clustering.

## Introduction continued

High Availability clustering aims to increase the availability of IT applications by reducing the impact of hardware and software faults - for instance if one server crashes, a backup server will take over from it and, ideally, end users will not notice any interruption in service.

High Availability (HA) clustering is typically used in circumstances where down time is costly - the main costs often being lost revenue or lost productivity from staff, or both.

High Availability clustering is a mature and well understood technology, although previously only available in the realm of commercial high end Unixes, it has in the last few years become feasible and beneficial for mid-market PC based systems running commodity operating systems (for example Linux).

Applications with a high business value are the best candidates for clustering as the cost of the clustering technology (hardware, software, administration etc) can be offset easily by the avoidance of downtime in the future - for example email servers, file servers, database servers and web servers.

To give some examples of where HA clustering is useful :

a) An e-commerce website would want to ensure that their site is available and fully working to the worldwide audience all the time. There may be a corporate requirement for 99.99+% uptime, which includes maintenance periods etc.

b) A factory has its production line managed by a PC server. Initially downtime wasn't a major problem, but as production has ramped up, downtime of even a few minutes can cost thousands in lost revenue.

c) Organisations may offer IT services to others. Use of a High Availability solution would allow them to offer a guaranteed service level agreement (SLA) - this would be most relevant for instance web hosting, data processing and application service provision.

With the growth of computing by the general public and business, demand for computing resources has risen. This is also coupled with the expectation that the resource will be available when the end user attempts to access it - day or night.

## Lifekeeper for Linux – Overview

LifeKeeper for Linux is a High Availability clustering application produced by SteelEye Technology Inc. (<http://www.steeleye.com>) that facilitates High Availability clustering by actively monitoring the underlying operating system (IP/Disk/System responsiveness) and applications (processes, response times etc).

In the event of a failure occurring LifeKeeper can either attempt to recover the resource locally (for example restarting a process) or it can move the resources across to another node within a cluster. LifeKeeper nodes also monitor each other, so in the event of a node failure, recovery of all applications on failed node can be undertaken.

## Lifekeeper History

LifeKeeper was originally designed and developed by AT&T Bell Labs to ensure high availability of their worldwide voice network system running on Unix-based Star Servers (~1990). The need to ensure the highest levels of availability for such a mission-critical system required a design focused on reliability, performance and efficiency, which are still core values of the LifeKeeper product today.

After AT&T divested the LifeKeeper division to NCR, SteelEye acquired the technology and brought on board the original development and QA teams, who today form the core of the Technology and Services resources. This depth of knowledge and expertise in high availability systems, storage management and resource monitoring is what differentiates SteelEye and its solutions.

LifeKeeper has over 10,000 licenses, and is one of the worlds leading technologies for High Availability Clustering.

LifeKeeper for Linux is currently supported on Asianux 1.0 and 2.0, Novell SUSE SLES8, SLES9, SLES 10 and Red Hat RHEL3 and RHEL4. Standard versions of Sendmail, MySQL, and Oracle are also supported. There is no requirement for you to purchase and deploy higher end versions of either the OS or applications. Development efforts are ongoing to support new hardware and operating system releases as required.

This wide range of operating system support ensure customers are not locked in to any one operating system, and have the flexibility to upgrade when required.

## Lifekeeper Core Software

The LifeKeeper Core Software consists of the following components:

### **LifeKeeper Configuration Database (LCD)**

The LCD stores information about the LifeKeeper-protected resources. This includes information on resource instances, dependencies, shared equivalencies, recovery direction, and LifeKeeper operational flags. The data is cached in shared memory and stored in files so that the data can be remembered over system boots.

### **LifeKeeper Configuration Database Interface (LCDI)**

The LCDI software queries the configuration database (LCD) to satisfy requests for data or modifications to data stored in the LCD. The LCDI may also be used by the Application Recovery Kit to obtain resource state or description information.

### **LifeKeeper Communications Manager (LCM)**

The LCM is used to determine the status of servers in the cluster and for LifeKeeper inter-process communication (local and remote). Loss of LCM communication across all communication paths on a server in the cluster indicates the server has failed.

### **LifeKeeper Alarm Interface**

The LifeKeeper Alarm Interface provides the infrastructure for triggering an event. The `sendevent` program (`$LKROOT/bin/sendevent`) is called by application daemons when a failure is detected in a LifeKeeper-protected resource. The `sendevent` program communicates with the LCD to determine if recovery scripts are available. If so, `sendevent` calls the `is_recoverable` and `recover` programs which then call the appropriate recovery scripts for the resource.

### **LifeKeeper Recovery Action and Control Interface (LRACI)**

The LRACI software determines the appropriate recovery script to execute for a resource. The shell version of this program is `$LKROOT/bin/perform_action`. The LRACI software invokes the appropriate restore / remove scripts for the resource.

## Application Recovery Kit Software (ARKs)

One of the key advantages of SteelEye's LifeKeeper® architecture is that neither system kernels or target applications need to be changed in order to provide application and data protection.

Instead of requiring changes to users' compute environments, LifeKeeper's Application Recovery Kits (ARKs) are used as an efficient and flexible mechanism to integrate LifeKeeper with the Operating System and Applications. The ARKS provide monitoring, recovery, start up and shutdown operations for the application or resource concerned.

## Application Recovery Kit Software (ARKs) continued

The core LifeKeeper product includes volume, file system, IP and the Generic Recovery Kit. Specific, supported, recovery kits are purchased as an additional extra. The Customisation Kit allows the user to create their own recovery kits, which can be tailored for any application.

Each Recovery Kit defines specific resource types associated with an application type that can be placed under LifeKeeper protection. Resource types are defined with corresponding software for the following:

### Resource Monitoring

This software monitors resource instances and notifies LifeKeeper when a failure occurs.

### Recovery Direction / Action

This software provides the information needed to direct the recovery of a resource instance. This includes a description of the resource dependencies so that LifeKeeper will know how recovery should be directed in case of failure and the basic actions that resource recovery may require.

Supported ARKs are available for the following Linux applications:

- Oracle
- My SQL
- PostgreSQL
- Sybase
- DB2 Universal Database
- Progress
- Informix
- Apache
- NFS
- Samba
- SAP
- Sendmail
- WebCT
- Blackboard
- Linux Print Services
- Logical Volume Manager
- Rational Clearcase
- Websphere MQ
- Software RAID
- LAMP
- VMware ESX Server Virtual Machine

(for a full list please see [http://www.openminds.co.uk/lifekeeper/applications\\_supported.htm](http://www.openminds.co.uk/lifekeeper/applications_supported.htm) )

There are some applications which LifeKeeper cannot support - generally this is due to Operating System constraints (for instance Real Time applications). It is also difficult to support applications which are bound to a particular server (whether by hardware or some other means e.g. Hostname).

### **Generic Recovery Kit**

A number of template recovery kits are included within LifeKeeper for Linux, such as the Application with File System Recovery Kit or Application with Disk Partition Recovery Kit. These recovery kits allow protection of a generic or "customer-defined" application that has a dependency on a file system or disk partition, but no associated recovery kit to define the resource. These kits allow a user to define recovery scripts that are customised for a specific application.

Recovery templates are used when a simple recovery is required that simply starts and stops an application.

### **Customisation Kit**

The LifeKeeper Customisation Kit provides a user-friendly way to develop integration between LifeKeeper core and any other application, enabling users or SteelEye partners to write their own custom recovery kits. Usually a custom recovery kit leverages as many of the existing LifeKeeper resource instances as it can.

The Customisation Kit is used when the application recovery is dependent on other resources and a specific behaviour is required to manage the recovery process.

For example, the Apache Web Server Recovery Kit supplies its own webserver/apache resource instance, but its job is merely to start and stop the web server. It creates dependencies of the LifeKeeper supplied comm/ip and gen/filesys resources to protect the IP addresses and file systems it needs to function correctly.

### **Components of an ARK**

Each resource instance consists of two separate components: an action component (for recovery and monitoring) and an administration component (for creation, extension, and removal). The action component performs start, stop, and monitoring operations for the application associated with a resource instance. The administration component is usually attached to a top-level resource instance and performs administration operation, such as creating the entire resource hierarchy, including the associated dependencies. The administration component also includes a GUI interface, which defines how the LifeKeeper Java-based GUI should collect input parameters for the administration components.

## Bespoke Applications continued

component is usually attached to a top-level resource instance and performs administration operation, such as creating the entire resource hierarchy, including the associated dependencies. The administration component also includes a GUI interface, which defines how the LifeKeeper Java-based GUI should collect input parameters for the administration components.

## Clustering Technology – What Clustering achieves

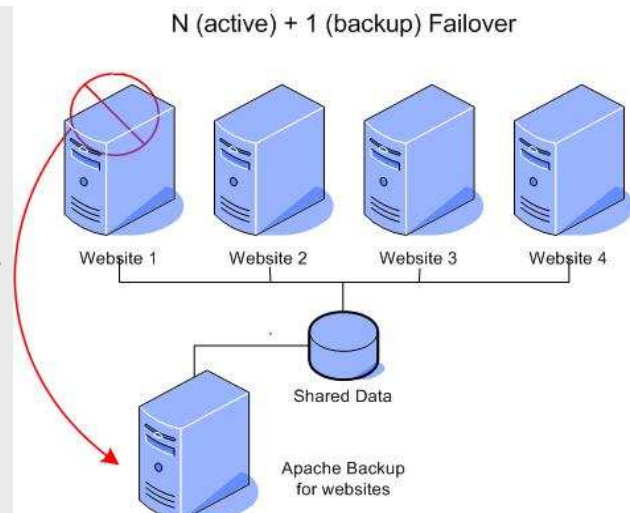
Clustering allows for high availability of an application/service to be achieved - in that in the event of a failure (e.g. Hard disk dies / Computer crashes etc) the service can be removed from service and put into service on another computer. This means that tolerance to failure is built into the system, hopefully resulting in no loss of availability of the service. The effect of this is that the service is always available to the end users and the value the service provides is maximised.

## How Clustering works

A cluster contains numerous (>1) computer nodes. Some nodes may be active (running a service), and some may be in a standby state where they can take over a service from a failed node. Other nodes may be both active, and acting as a standby for another server, this leads to there being three main configurations for a HA cluster - namely Active/Active, Active/Standby and N+1. These configurations are discussed in greater detail below.

## Cluster Configurations – N+1

It is possible for a cluster to be of N+1 size, where N nodes are active, and 1 node is used for a backup in the event of any (or all) of the N nodes failing. This configuration can reduce hardware costs, while giving redundancy. An example with the Apache web server is shown.

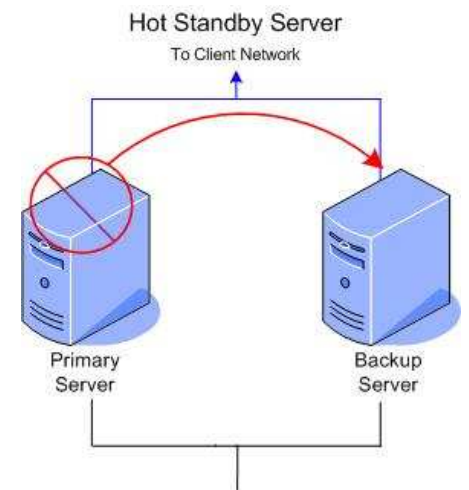


In the event of the server hosting “Web Site 1” failing, in any particular way (e.g. Network, disk, software or power failure), “Apache Backup Server” can take over hosting of the website. Should another server also fail, then “Apache Backup Server” can also take over as well.

## Active/Standby

This configuration dedicates a backup server to standby in case the primary node fails.

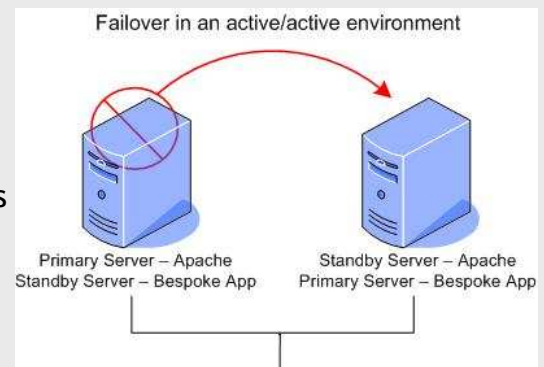
Active/Standby ensures that performance does not suffer in the event of a failure on the primary node, but has the disadvantage of requiring a clone of all hardware to act as backup. Costs can be reduced by having a less powerful (and therefore less expensive) computer acting as backup if necessary.



Often our customers will use the introduction of LifeKeeper as a means to improve system responsiveness by upgrading the existing solution's hardware, and to use the old hardware as the backup node.

## Active/Active

If we were to always have one active computer, and one (or more) computers waiting in hot backup, there would be a significant waste of resources occurring. One method around this is to use an Active/Active configuration where any one server is capable of running multiple applications if a failure of another server occurs. There may be a performance degradation in the event of a failure, but this may be deemed to be acceptable.



## Disaster Recovery Solution

The requirement for an off site, real time recovery solution is becoming more important with the increasing globalisation and distributed nature of organisations. Insurance policies are now requiring remote backups, where there must be a considerable distance between nodes.

Depending upon the value of the data or service involved, there can be multiple different recovery strategies - ranging from a from scratch approach - ordering new hardware/restore from backup, to something more complex involving semi-automatic or fully automatic fail over. The more basic the restoration plan is the greater

## Disaster Recovery Solution continued

time will be required for it to complete, and the greater loss will be incurred to the business - conversely the more complex the plan the greater resource requirements there will be (e.g. Real time mirroring requires a point to point link between the two sites, and extra hardware).

Having an off site backup isn't enough to ensure continuation of operations in the event of a site disaster - there may be considerable data loss between the last scheduled backup, and there is a requirement for operator intervention to bring the service back up and running.

LifeKeeper based solutions provide for automatic (or semi-automatic) fail over, with real time replication of data between primary/standby servers. The combination of these allow for a near seamless migration of the application, ensuring that your business keeps running.

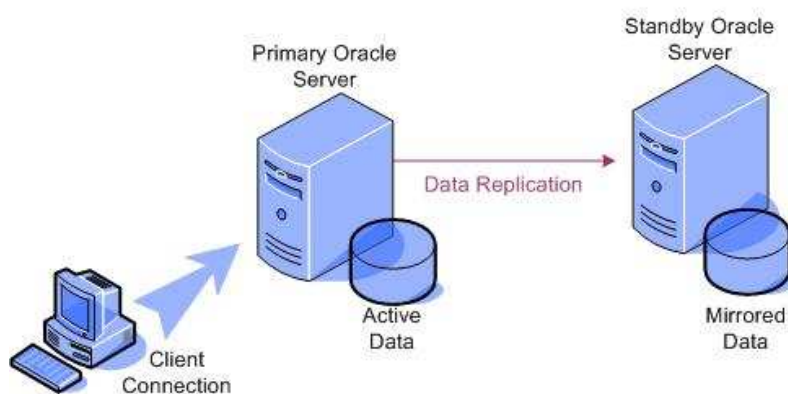
When considering Disaster Recovery solutions, it is often most affordable and practical for at least one of the server nodes to be located near the largest body of clients - therefore usually one server will remain in a company office, and the new additional one will either be located in a remote office, or alternatively in a remote hosting company.

As mentioned earlier, SteelEye's data replication software is able to replicate data between two nodes. It is also possible to pause the mirror at the remote end, take a backup copy of the data, and then resume the mirror - providing flexibility and total data security.

In order for there to be an off site recovery solution, there needs to be some sort of virtual LAN between the protected servers, so network routing can still take place in the event of one server failing. One way of doing this is to use a Virtual Private Network (VPN).

By their nature, there is a higher risk of there being a split brain when using a geographical disparate configuration, as it is normally not feasible to have numerous redundant links between nodes. To counter this, it may be necessary to disable remote fail over without human intervention in case there is network disruption between sites.

### 2 – Node Disaster Recovery



## Virtual Servers

The use of virtual servers to host business critical applications is growing rapidly as IT organizations experience the significant cost savings and optimised resource utilisation through server consolidation. It is not well understood, however, that the use of virtualization technologies can decrease the availability of applications and the services which they host. The need for integrating an automated high availability clustering solution to monitor and protect virtual servers becomes evident when the increased risk of failure is understood.

The potential for application failure is increased due to two factors. First, virtualization increases the scope of failures as more applications and services are placed onto a single physical machine. If this consolidated server should suffer an outage, multiple applications and business services will be unavailable. Second, the primary causes of application outage are software related; the more complex the software stack being used by an application, the higher the risk of failure. The deployment of a virtualization layer increases software stack complexity and increases the risk of your applications suffering from outages.

Ensuring high availability and disaster recovery for your virtual machines is therefore just as, if not more, vital to achieve business continuity.

For more information please visit: <http://www.openminds.co.uk/virtual.html>

## Service Virtualisation

Above all else, we want as few of the end users as possible to be aware of a fail over, and we certainly don't want them to have to reconfigure their client applications (e.g. changing host name for their email client, or typing a different address into a web browser).

An essential part of high availability clustering is **service virtualisation**, which in a nutshell describes the ability of a service to be run on any node within the cluster, without the end user being aware of which node it is running upon. In order for this, it is necessary for all resources the application depends upon (it's dependencies) to be available to all nodes within it's cluster. The resources which need to be available are usually IP address(es), storage (data files), configuration files (for settings) and binary program files. Some of these resources may be shared between nodes on shared storage (e.g. data files) while others may be installed locally on each node (e.g. binary files).

## Significance of Virtual/Floating IP addresses

The traditional method of configuring a server is to use DNS “CNAME” entries for each service, and only one IP address per node regardless of how many services it is running.

For example, a small mail server may have two DNS CNAME entries - “pop” and “smtp” for the same IP address (192.168.0.1) in an office. If the email server becomes too busy, and an additional machine is added to take some of the load away, one of the services would move onto the new server, and the DNS entry would be updated respectively. In this situation, clients would not need to reconfigure, as the host names used would not be changing and the DNS records would have time to update and propagate throughout the network overnight.

The scenario described above is not suitable for High Availability clustering, because it may take many hours for a client machine to realise the DNS entries have been altered. Therefore in HA clustering, we use a separate IP address for each service, and instead move the IP address from one node to another as required.

The benefit of moving IP addresses from one node to another is that clients will often be unaware that a fail over has occurred (many applications will silently retry if they lose their connection momentarily), and if a stateless protocol is being used (e.g. NFS over UDP) then there is no state lost if the communication link is broken and then reformed between client and server.

Unfortunately most communication is done over the stateful TCP protocol, connections will therefore become broken when a fail over occurs (because the underlying MAC address used on the network cards will have changed). It is necessary to break the connection anyway, so telling the client machine that any it will need to re-initialise with the server.

## I/O Considerations

In order for an application to move from one node to another, without losing data, it is necessary to make the underlying data available on all nodes in the cluster. There are a number of ways of doing this, depending upon whether it is acceptable to suffer from limited data loss or not.

## Shared Storage

This is the most frequent configuration, where a number of nodes all have access to a (normally SCSI) disk array or SAN. This provides for the greatest flexibility in cluster size, however it normally places restrictions on the distance between nodes. It is also generally the most expensive solution, but can offer the greatest features and reliability (e.g. RAID 5 offers mirroring and performance).

## Data Replication

Data replication is a software based solution which is capable of maintaining data synchronisation between two nodes over an Ethernet link. Generally, replication is a one way process, where the standby node is the target of the mirror and the data is inaccessible. The mirror source is accessible, and changes to the source are mirrored to the target server when they occur.

Currently SteelEye's data replication software supports a one to one mirror configuration only, however one to many replication is currently under development.

As data replication takes place over Ethernet, it has the advantage that the servers can be in different geographical locations, making it an ideal solution for disaster recovery.

It can also be used as a replacement for shared storage, as it is often cheaper, and requires less hardware. Data replication allows for generic white box PC's to be used instead of the more expensive servers.

With data replication, reads of the data will be undertaken locally, while writes will occur on both nodes.

There are two general forms of data replication that LifeKeeper provides, Synchronous and Asynchronous mirroring.

### Synchronous Replication

Synchronous Replication writes data to the local server, then to the remote server, and once this is done it finishes the write and returns to the application - because it is writing data twice and one of the writes is over a network connection a high bandwidth, low latency link is ideally required between servers, otherwise performance will suffer.

### Asynchronous Replication

Asynchronous Replication this is similar to the synchronous replication above, but the local system does not need to wait for the remote system to write data to acknowledge receipt of data before returning to the client return. This provides greater performance benefits, allowing the local system to send batches of writes. It also provides the ability for the mirror to be broken at quiet times so allowing for a traditional style backup to take place on the standby server, and then resumed without having to undertake a full resynchronisation.

Asynchronous replication by it's nature brings with it a greater risk of data loss if the primary node crashes midway through a write, and customers need to determine whether they want the reliability of synchronous replication or the speed of asynchronous replication.

## Asynchronous Replication continued

Generally Asynchronous replication is recommended for remote site replication, where traffic will be going over a higher latency / lower speed link (WAN).

## Bandwidth Requirements for Data Mirroring

Bandwidth requirements for data replication is a frequently asked question. To answer this, some idea of the volume and quantity of writes to a disk needs to be known, along with the type of mirroring to be used, and if there are any performance requirements or limits to how much data can be potentially lost.

If using Asynchronous mirroring, then greater performance will be observed for clients using the active server, however over a slow link (or on a busy server) the write queue can end up holding a relatively large amount of data. The write queue length is tenable, however the larger it is, the greater scope there is for data loss.

Because data replication needs to copy data to a remote location, it ideally requires a high bandwidth, low latency link between nodes. If it is being used as a replacement for shared storage, and both nodes are in close proximity, fast Ethernet or gigabit Ethernet is normally adequate.

## I/O Fencing

Having replication, or shared storage on it's own isn't quite enough for a HA cluster, as there needs to be some means of controlling access to the data to avoid two (or more) machines from editing the same files at the same time - which would result in corruption of data. In order to protect the data on volumes, all HA clustering solutions provide some sort of protection to ensure that only one node can access shared I/O resources at any one time.

Solutions to I/O fencing, generally rely on whether the file system is mounted or not, or may also use some sort of STONITH device to power cycle a non-responding server.

However, this configuration is not fool proof - it is possible for an administrator to do what he/she shouldn't, causing data corruption. SteelEye's LifeKeeper has arguably the best solution to I/O fencing, in that it uses a Native SCSI Primitives (aka SCSI reservations). These allow a server to maintain an exclusive handle on a SCSI LUN, and can therefore protect the volume at a low level from corruption. In the unlikely event of a split brain scenario (more information on this is given later on) when one node loses it's exclusive handle on the storage, it will immediately shutdown, thereby guaranteeing data integrity.

## I/O Fencing continued

Unfortunately SCSI reservations are only available under Linux when used with Shared Storage. With all other configurations it is necessary to use STONITH devices, which are fully supported by LifeKeeper. In a nutshell, if a node loses contact with another, one node will use the STONITH device first to reboot the other. (STONITH is short for Shoot The Other Node In The Head).

A further discussion of the behaviour of nodes in a split brain style situation is included in the “Split Brain Protection” discussion below.

Other mechanisms for I/O fencing, involving proprietary file systems do exist, however these also add complexity to a solution, and have an impact on the operation and administration of the cluster.

The only draw back for I/O fencing through SCSI reservations is that the choice of SCSI controller may be limited to one supported by SteelEye, however most mainstream controllers are supported and certified by SteelEye.

SteelEye have undertaken considerable development on the Linux kernel to enhance and fix the SCSI reservation process. These changes have been released back into the community.

## Clustering Operation - Overview

This section discusses how clusters of computers are formed, why clusters are used and what are typical configurations for a cluster.

At the simplest level, clusters can be built from generic white box PC's with no special hardware. However, in order to provide the maximum availability of the application we need to enhance the reliability of the system as far as reasonably possible, so it is worth considering hardware which has some inbuilt tolerance (EEC registered memory, multiple power supplies, RAID based storage) and redundancy (e.g. multiple network interfaces) or even lends itself to ease of serviceability (e.g. Hot swap fans, disks, IO controllers). Most commercial servers have such features, and can be purchased from companies like HP, Dell or IBM. LifeKeeper is supported on all major hardware (HP, IBM, Dell and Sun).

When designing / detailing the specification for a cluster, it is important to ensure that there is no single point of failure, so multiple paths should be maintained where ever possible (e.g. Redundant power supplies, redundant network paths, redundant I/O paths and so on). Some redundancy can be carried out automatically within hardware (EEC Memory), and some can be undertaken through a combination of software and hardware (MultiPath I/O, Network interface bonding/teaming).

## Load Balancing

High Availability clusters do not lend themselves to automatic load balancing like a performance cluster (e.g. OpenMosix) is able to undertake. Load balancing with HA clusters occurs in a manual manner, in that applications will be spread through out nodes in the cluster, and if possible resources which are excessively busy will be split into smaller sub units which can be distributed across nodes within the cluster.

Some load balancing can be undertaken using the Linux Virtual Server software, which performs simple load balancing between nodes, in a similar manner to hardware load balancing routers. However this requires that the underlying software applications be aware that the data it uses may change due to another servers operation, which has issues for data caching and consistency.

High Availability clustering solutions are able to operate in active/standby and active/active modes, (mentioned earlier). The active/active configuration allows for greatest distribution of applications, and therefore greatest utilisation of processing power across nodes. The active/standby configuration normally entails one machine standing unused ready to take over in event of a failure on the primary node - this has the advantage that the backup machine will offer good performance (as it is doing nothing else) however it does entail a significant extra expense in hardware. A solution to this is to use an N+1 cluster configuration, where there are N active nodes, and one standby server which is configured to act as standby for all the applications running on the N nodes; of course in the event of two failures (which is probably unlikely) the single backup node may well be overloaded, and performance may suffer.

## IP Availability

LifeKeeper monitors network interfaces, and checks that it is possible to reach each IP address. This helps ensure that the application is reachable by clients and working correctly.

As mentioned earlier, the use of “floating” IP addresses results in end users not having to reconfigure their client settings, so failovers can take place with minimal interruption.

LifeKeeper is able to check that a node is on a connected network segment, and can perform recovery should one node become disconnected from the LAN.

## Resource Hierarchies

As mentioned above, in the Server virtualisation section, LifeKeeper attempts to make available the same resources on each node within a cluster for an application, allowing the application to move from one server to another. Naturally some resources depend upon others - for instance a SQL server process requires an IP address, a file system (data files) and so on. Some of these dependencies may depend upon others - for instance the IP address will depend upon there being a valid network, and the data files will depend ultimately upon a physical disk being available.

## Resource Hierarchies continued



The above image shows a Microsoft SQL Server hierarchy which is under protection by LifeKeeper. The “MS-SQL” resource represents the SQL Server process family (there may be multiple processes protected under a resource - for instance the SQL server, the SQL Search agent and suchlike all represented by one resource icon). The MS-SQL resource depends upon two other resources - namely an IP address (192.168.20.45) and a volume (SQLVOL.G). When taking this screen shot D1 was actively running the SQL server instance (hence Active on the squares in D1's column) and the hierarchy was in a protected state (i.e. Fail over to D2 could occur) denoted by the green discs. D2 is acting as a standby, and would be the server to take over from D1 in the event of D1 failing. If D2 failed the SQL hierarchy would be unaffected, unless D1 also failed.

The heartbeat like symbols on the D1 and D2 icons, illustrate that multiple communication paths exist between nodes, and are operating correctly.

## Failover Mechanics

How does LifeKeeper monitor the cluster to determine when a fail over to another node is required?

As mentioned previously, LifeKeeper monitors the resources in a cluster as well as the status of the node itself (by comparison some cluster solutions only monitor the health of the node itself).

As well as checking the status of resources on a node, LifeKeeper is also configurable to take action on deliberate powering down of a node; or when a node fails. LifeKeeper is also configurable as to what happens when a node fails (fail over) and what it does when a node comes back on line (should it switchback resources to a node?). There are some cases where fail over is not desired (for instance if

## Failover Mechanics continued

undertaking a disaster recovery solution where there is a higher chance of a split brain scenario occurring than a leased line failing - it is therefore possible to stop an automatic fail over from taking place without administrator intervention.

## Intra Cluster Communication

Each node of a LifeKeeper cluster should have at least two communication paths with other nodes - normally provided by one dedicated Ethernet connection, and one shared connection which may be the public LAN connection to the server. A RS232 serial connection between cluster nodes can also be used, although this doesn't scale as well as switched Ethernet, but is a cheap and ideal solution for 2 node clusters.

By having redundant paths between servers it is possible to ensure that there is not a single point of failure in the cluster.

LifeKeeper communicates between nodes over these links, helping to ensure that LifeKeeper is running and that the nodes are alive. In the event of a node failure, one of the remaining nodes in the cluster will detect it, and initiate recovery of the failed applications.

By default LifeKeeper will wait 15 seconds (3 tries with a 5 second gap between) before initiating a fail over when a host does not respond. This is configurable to a longer or shorter duration. A longer duration may be required if the servers are excessively busy or geographically distant. A shorter duration can be used if the nodes are close to one another and the risk of false failure detection is minimal and you wish downtime to be minimised.

As mentioned earlier, the Application Recovery Kits (ARKs) include tests to check whether the application is actively running. Generally there are two tests undertaken, the first is commonly called the "quickCheck" which may check that a process is running and perhaps listening on a particular port.

The other script (which is optional), commonly referred as the "deepCheck" script, undertakes a much more thorough check and is generally scheduled to run less often than the quickCheck script (because it may take longer to run, and require more resources). The deepCheck script may attempt to execute queries on a database, or request particular web pages from a website.

In the event of a failure of a resource, LifeKeeper can attempt to undertake a local recovery (if configured to do so) or it can undertake a fail over to a remote host. This is configurable when creating the resource hierarchy.

## Resource Checking continued

Due to the relationship between resources in a hierarchy, if one of the dependent resources fails, so too will the parent, triggering a fail over of the entire hierarchy.

## Failover Times / Performance

The time taken for an application to recover generally depends upon the application itself - in that some application may have to roll back through a transaction log file (in the case of a database), while others (e.g. Web servers) can start quite quickly.

The time needed for LifeKeeper to recover a resource is generally quite low (at most a few seconds), most of the time will be spent at the operating system level bringing resources into service (e.g. Mounting file systems, checking disk integrity, setting up IP addresses and performing disk I/O when loading binary application files).

## Distributed Cluster Meta Data Storage Model

One of LifeKeeper's strengths is that it does not require a separate Quorum partition for storage of cluster meta data, instead the cluster members synchronise between each other using a light weight peer to peer protocol. This provides for greater resilience and flexibility within the cluster as there are multiple copies of configuration on each node, no one partition is a single point of failure, and the nodes do not need to be attached to a physical volume.

Some other clustering technologies using Quorum devices do allow for clustering over a WAN with the quorum synchronised via a form of replication software, aside from being a single point of failure and prone to corruption, it makes the overall solution more complex than necessary.

LifeKeeper's data storage model removes the requirement for a separate partition / LUN for intra node communication. LifeKeeper's storage model also allows any node that is able to talk to any other, to be part of a cluster, hence clustering over a WAN/ LAN is possible, and software based data replication storage can be used.

## Split Brain Protection

A split brain scenario is one where all nodes believe the others are unavailable/off line.

It is often caused by inadequate resilience of communication pathways between nodes, and the result of there being a single point of failure (e.g. A single router or switch common to all network interfaces being off line).

It is advisable to avoid the risk of a split brain occurring as far as

## Split Brain Protection continued

practically possible - for instance ensure there are no single points of failure between nodes (different paths, different chip sets used in hardware devices etc).

The following scenarios illustrate what can happen in a split brain scenario -

### Active / Passive Split Brain with Shared Storage

If all of the heartbeat cables are removed, a split brain situation will occur. Both servers will try to get control of the resources which were in service on the other. One will get control and the other one will panic/reset (to avoid data corruption) upon realising it has lost it's SCSI reservations.

When a split brain occurs, it is really a timing situation. Whichever node "wins" getting the SCSI reservation first is the one that gets the resources. The node that wins is usually the node with the resources active, but not always.

### Active / Active Split Brain with Shared Storage

Again, removing all of the heartbeat cables will cause a split brain situation. Both servers will try to get control. A Race condition will occur and one or both systems will be rebooted. One system would end up getting all of the resources.

### Active / Passive Split Brain with Data Replication

If a split brain situation occurs, both servers will have the LKDR resource in service. When this is detected (e.g. The other node becomes accessible), data corruption is avoided by requiring manual intervention to get the resynch started again. An error message stating this gets logged in the system log. The end user will have to decide which side of the mirror is the best and take the resource out-of service on the server with the "bad" mirror. This will start the resynch from the "good" side to the "bad" side.

### Active / Active Split Brain with Data Replication

This is the same as an active/passive split brain.

In the event of a split brain occurring when using data replication, it is likely that one node will have become cut off from the network, while another is still accessible from anywhere - in which case the decision of which data to keep should be relatively easy. If both nodes are cut off from the network, even though they may put the resource into service at both ends, no end users will be able to modify the data, so it would not matter which copy of the data the administrator chose to keep.

## Active / Active Split Brain with Data Replication continued

As you can see, the Split Brain scenario is avoided / minimised through the use of multiple paths for communication between servers.

If a split brain scenario does occur it is most likely to happen between remote nodes which are using some form of replication between each other (e.g. SteelEye's Data Replication software).

If this does occur then it is necessary for the administrator to decide which copy of the data is most accurate and to resolve the problem. In some situations (e.g. Using Microsoft Exchange) it is not recommended that remote fail over is automatic - this removes the chance of a split brain from occurring, and ensures data integrity.

## Management

LifeKeeper can be managed in two ways, either through the command line (useful for remote administration over low speed links) or via a graphical user interface (GUI) which is cross platform and intuitive to use.

The command line interface is only available for the Unix variant. All platforms are able to use the GUI for administration.

The GUI can be launched from either the command line or a web browser, making it accessible from any client as long as the Java Run Time is installed locally.

The following screen shot shows the GUI where there is a two node cluster. One node is actively running the Apache web browser, while another is acting as standby.



The left panel shows the resource hierarchy, where a tree like structure is used to illustrate the dependencies in the hierarchy.

Each resource in a hierarchy will either contain a user inputted tag (entered upon hierarchy creation) (e.g. ip-192.168.20.44-www) or a default tag (e.g. export/apache). In the hierarchy above the ApacheWebserver resource has two direct dependencies - an IP address and a file system. The file system in turn depends upon a block device and a physical disk.

The screen shot also shows the fail over order, where a numerical value is assigned to a hierarchy on a server to determine the order of fail over. By default the server with a value of "1" will be the active server, and the fail over nodes will be in order of increasing numerical value (10 next, then 20 etc). The values are changeable at any time, and can be changed to alter where applications are actively running.

In the event of one of the nodes failing suddenly (in this case D1), the following is can be seen :



When this screen shot was taken, LifeKeeper was unable to contact D1 to determine the state of it's resources. After attempting to contact it three times, it will initiate recovery on the second node (D2) as shown below -

## Management continued



A couple of things to note -

Firstly the green heartbeat line through the icon of the node D1 has changed to a red flat line - this indicates that the node is off line, and unreachable.

Secondly, the resources which are being brought into service on D2 are no longer marked with green dots in the hierarchy tree - the new icon indicates that the resource is in service, but unprotected.

A few seconds later the following screen shot has been taken -



## Management continued

This illustrates that fail over has successfully ended, and that the entire resource hierarchy is unprotected (as we would expect when one node of a two node cluster is unavailable).

After the fault with D1 has been resolved, D1 comes back on line and the following is seen :



As can be seen by the icon for D1, it is now back on line and running LifeKeeper. D1 is also now providing protection for D2 in the event of a failure on D2.

By default LifeKeeper will not “switch back” the resources to D1 automatically, instead it is assumed that an intelligent decision will be made by the operator - this is useful for instance if D1 is “bouncing” while under going maintenance.

In order to minimise disruption to end users, it is normally best to allow an operator to manually move the resources back into service on the primary node when the service is less likely to be busy.

As mentioned earlier, the “switch back” behaviour is configurable, so if necessary D1 could automatically run Apache when it came back on line, such a configuration may be necessary if the backup node was insufficient to run the service to an acceptable level of performance for long periods of time.

## Acknowledgements

The following are trademarks of SteelEye Technology Inc

- SteelEye LifeKeeper
- SteelEye Data Replication

The following are trademarks of Microsoft Corporation

- Microsoft Windows (2003/2000/NT)
- Microsoft Cluster Server
- Microsoft SQL Server
- Microsoft Internet Information Service (IIS)

The following are trademarks of Sun Microsystems

- Solaris (x86/Sparc)

The following are trademarks of RedHat Linux Inc.

- RedHat Advanced Server (AS)
- RedHat Cluster Suite

The following are trademarks of SuSE Linux Gmbh.

- SuSE Linux Enterprise Server (SLES)
- SuSE OpenExchange

The following are trademarks of Intel Corporation

- Intel Pentium (II, III, IV)
- Intel Xeon

The following are trademarks of Hewlett Packard (HP)

- HP ProLiant
- HP OpenView

Other trademarks should be acknowledged. Please report omissions, or updates, to [info@openminds.co.uk](mailto:info@openminds.co.uk) quoting the document name and revision number.

The most recent version of this document should be available from <http://www.openminds.co.uk>

## About Open Minds High Availability Solutions Ltd

Open Minds High Availability Solutions Limited provides support, training and consultancy for High Availability IT deployments.

Founded in 1990, Open Minds focus is in using their experience of delivering high availability and server recovery to serve their customers. Open Minds solution architects have over 10 years experience of LifeKeeper and other High availability products on various platforms, including NT, Window 2000, Unix and Linux.

Using a unique network of partners and products, Open Minds can plan and implement systems availability on a simple two-node cluster on a LAN, to a sophisticated multi node, multi-application disaster-recovery across a WAN.

Open Minds works with its customers and partners to create systems and application availability scenarios in line with their business objectives.

## About SteelEye Technology Inc

SteelEye Technology Inc (<http://www.steeleye.com>), is a leading provider of IT solutions for business continuity and disaster recovery.

The SteelEye LifeKeeper family of software products and services offer unique scalability in terms of integrated solutions for data protection, high availability clustering, and wide-area disaster recovery on Windows and Linux.

SteelEye LifeKeeper enables enterprises of all sizes to ensure continuous uptime of business-critical systems. With 'out of the box' support for the widest range of applications, databases, and storage subsystems running on Intel-based Windows and Linux servers, SteelEye LifeKeeper ensures enterprise-grade reliability at a fraction of the cost and complexity of traditional solutions.

SteelEye LifeKeeper is proven in the world's most demanding business environments. Today, Global 1000 companies rely on SteelEye LifeKeeper to keep their systems running and their people productive.

## About This Document

Original author - Open Minds High Availability Solutions.  
Revision - 5.0  
Last updated - September 19th 2006.



Portions of this document are used under permission from SteelEye Technology Inc.  
Please send any feedback, questions or criticism to [info@openminds.co.uk](mailto:info@openminds.co.uk)