

# All-fix kits

Don't be daunted by LifeKeeper's modular approach – its flexibility and versatility make highly available clusters easy

BY MIKE JAMES, STORAGE EDITOR

LifeKeeper is best described as a suite of products on Linux and Windows that provide high availability. They do this by using one or more backup machines to take over should the active server fail. LifeKeeper takes a modular approach to creating a high-availability cluster that can seem overwhelming at first, simply because there are so many choices. The advantage is that it's very flexible and once you have mastered the ideas behind this suite, you can apply them in various situations quite easily.

The key component is the LifeKeeper core. This module keeps checking that the active server really is active and promotes a passive backup server to active duty if it is not. All of the servers in a cluster must have a method of communicating with one another in addition to being connected to the network that they serve. These extra communications paths are used to send heartbeat signals that the cluster can use to determine that the active server is working. You can use an additional TCP/IP connection, a serial link or a shared disk drive to transmit heartbeats. To avoid the problem of incorrectly assuming the active server has failed, more than one heartbeat path is used.

Detecting when the active server has failed and switching to an alternative server is the main task of the core module, but to allow the newly activated server to take over in a transparent way more has to be done – this is where the add-on recovery kits come in. Essentially, the recovery kits have two functions. First, they make the failover transparent to the user for a specific aspect of the server's behaviour. Second, they monitor individual applications and services within the server. The two most basic recovery kits are the IP and DNS recovery kits, which are part of the core package.

IP recovery involves transferring the IP address of the active server to the newly promoted server. When an IP resource fails, LifeKeeper attempts to bring it back into service using the same adaptor. If this fails, it tries to use a backup adaptor on the same server and only if this fails does it transfer the address to a backup server. If the backup server is in a different sub-net,

the IP address cannot be transferred in this way. The DNS recovery kit automatically updates the DNS server record to reflect the new IP address of the server following a failover.

There is more to do, however. Unless the backup servers are kept in sync with the active server, only a trivial sort of failover will go unnoticed by most users – even a print server has a print queue to maintain.

There are two ways of ensuring the same disk data is available to all the machines in the cluster. The first is to use shared storage – shared SCSI host adaptors or a SAN. This allows any of the clusters to take over the data, but you need to be careful not to introduce a single point of failure into the system. The alternative is to use storage local to each machine and use replication to keep everything up to date. This is what SteelEye Data Replication does. It creates mirrored volumes on the servers in the cluster. Only one volume is accessible to the outside world, the rest are for backup only.

Replication can be synchronous or asynchronous. Synchronous replication ensures the backup mirror copy is always up to date but the write operation isn't signalled as complete until both drives have finished the write. This ensures that a transaction cannot be lost, even in a failover situation, but it can reduce performance. With asynchronous replication, write commands can be queued on the mirrored volumes but the active server is allowed to continue processing as soon as it completes its write operation. Asynchronous replication may be less secure but it can be good enough for many situations where the possibility of some data loss can be tolerated.

## The real world

So much for the theory. How easy is it to set up and manage a cluster? The answer is that once you have the basic idea it's all very simple. The LifeKeeper GUI is a Java-based management console that can be run locally on one of the cluster servers or in a web browser on any connected machine. You simply add servers to form the cluster. The first task is to set up communication paths

between them so that heartbeat signals can pass. After this, the cluster is configured by creating resources on the active server and then extending them to the cluster (**Figure 1**). For example, to create a protected IP address that can be failed over as part of a server recovery, you first create an IP resource on the active server. In doing this, you define the IP address to be used and how it will behave when there is a problem, which network card it is associated with, and so on. After you have created the resource you extend it to other machines in the cluster so that they use that IP resource if failover occurs.

Once you have this basic idea it's easy to see how to define and extend all types of resources – DNS names, file shares, databases, and so on. What is less obvious is that you can extend each resource on an active server to different backup servers, so providing N-way recovery. For example, during a failover you could pass the file share support to one server and application resources to another for load balancing.

Moving on from the general facilities that the core module provides, there are recovery kits for a range of standard Windows and Linux applications, including Exchange, SQL Server, MySQL, Apache, Oracle and SAP NetWeaver. Each recovery kit is installed on top of the core module and adds to the basic facilities. The recovery kits know how to configure the resources that should be shared in a cluster and how to detect a problem failover with as little fuss as possible. For example, with the Exchange recovery kit you can quickly create resource hierarchies that correspond to a complete Exchange server using the wizard-driven



**Figure 1: Once a resource has been created it can be extended to other machines in the cluster so that they can take it over if the active machine fails**

interface and then extend this to other servers in the cluster. Of course, you have to have installed Exchange on each of the cluster servers, but the Exchange recovery kit makes it easier to weld these into a high-availability group and to manage them and the way that they fail over.

The most that a user should notice during a failover is perhaps the interruption of a request to view some mail item which then works again at the second attempt. The Exchange Recovery kit also makes it easy to create an N+1 cluster consisting of N active different Exchange servers with one backup machine for any server in the cluster.

While the configuration offers economy, it doesn't protect against two or more failures. You can, of course, test the configuration by forcing a manual failover. Once a server has been activated it generally stays in control until an administrator restores the original server to the active role – but automatic switch back is also available.

## Custom kit

If there isn't a recovery kit for an application that is important to you, the facilities provided by the core module make it likely that you can create a failover system for a custom application easily. You can create a Generic Application resource hierarchy that uses custom scripts to perform the basic functions required – restore, remove, quick check, deep check and local recovery. Templates for these actions are provided in Perl and VBScript on Windows and Perl and Bash in Linux. In most cases extending high availability to custom applications should be relatively easy.

You can also use the replication facilities to create a disk-to-disk backup of a server to another machine. Installed as a separate module as an add-on to basic volume replication, Disk to Disk Backup provides realtime backup with easy access to backup data and the ability to archive to tape. As well as continuous backup, periodic or scheduled backups are available.

LifeKeeper uses standard off-the-shelf hardware and, as long as they are all powerful enough for the job, there is no need for all the machines in a cluster to be identical. The result is a very flexible system allowing the creation of servers with potentially multiple backups. If this sounds complicated, remember that most installations will be simply two machines with standard application recovery kits – but even here there is the potential for extending the configuration to something more sophisticated once you are happy it's all under control. Sophistication and flexibility are easier to work with if a step-by-step approach is possible. LifeKeeper offers an excellent and general approach to building highly available systems. <

## System requirements

**Servers and OS** x86-compatible servers configured with Windows 2000 Server (Professional, Advanced or Datacenter editions) with SP 3 or later, or Windows 2003 Server (Standard, Enterprise, Datacenter or Web editions); or 32-bit or 64-bit servers with Red Hat or SUSE Linux

**Network communication** TCP/IP based  
**Memory** 128Mb+  
**Disk space** 55Mb+

## UK supplier

Open Minds  
 Tel 0845 345 3943  
 E-mail sales@openminds.co.uk  
 Web www.openminds.co.uk

## Price

**LifeKeeper for Windows**  
 From around £2,000 per node  
**LifeKeeper for Exchange** From around £3,000 per node

## Bottom line

**Pros** Modular approach brings simplicity without sacrificing power and flexibility.

**Cons** No 64-bit support for Windows yet (due in November). Number of options can make it a challenge to get started.