

# Inside Microsoft Cluster Server

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

1997 was a landmark year with respect to high-availability solutions for Windows NT: A slew of independent vendors released products that make applications more robust in the face of system failures, and Microsoft released its long-awaited clustering solution. Microsoft Cluster Server (MSCS), formerly code-named Wolfpack, is likely to be the dominant player on the clustering scene, which is already crowded with solutions. Two things differentiate MSCS from the crowd. First, it has been codeveloped with input from a number of big-league players, including Tandem, Digital, and IBM. Second, it is a standalone clustering solution and an open platform for the development of applications that can take advantage of the fault-tolerance the MSCS infrastructure provides. Third-party clustering solutions usually attempt to work with off-the-shelf applications. These applications don't realize they're executing on top of failover capabilities, a situation that can limit effectiveness.

Because MSCS has an extensible architecture for building cluster-aware applications and enjoys wide industry support, it is the de facto clustering standard for NT. More and more enterprise-level applications will be designed to be MSCS-aware, and as a systems administrator or developer, you are likely to cross paths with MSCS in the near future—if you haven't already. In this column, I'll define clustering, take a look inside MSCS, and describe how it works.

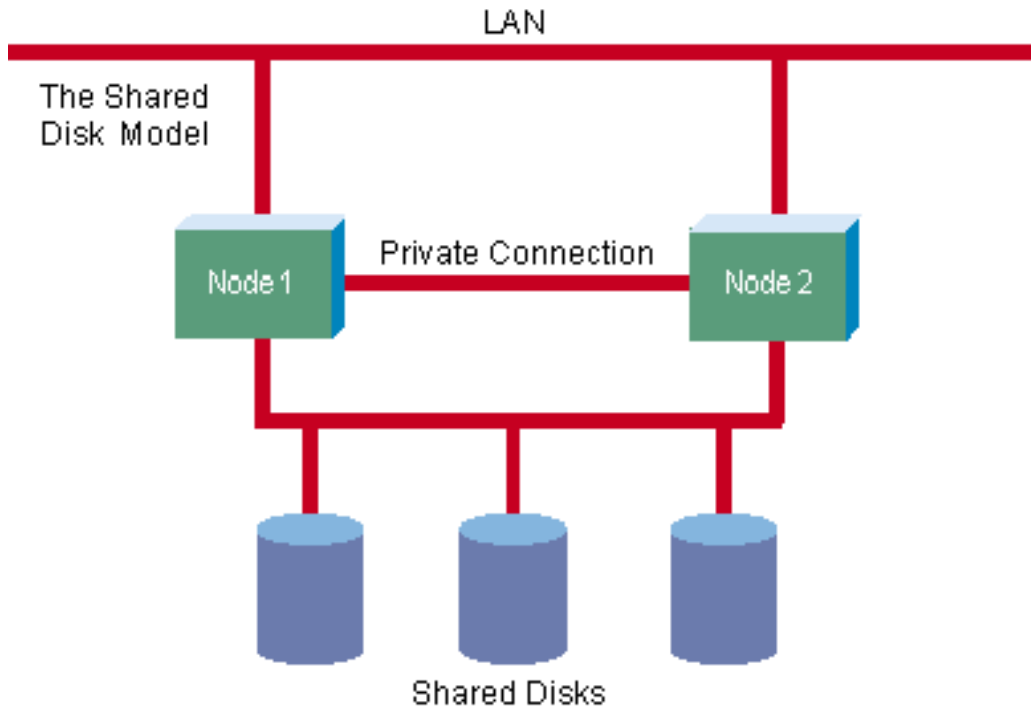
## Introduction to Clustering

In clustering, two or more servers work together to service a group of tasks, provide fault tolerance, or offer scalability (see Mark Smith, "Clusters for Everyone," June 1997). To provide continuous availability, a cluster must include at least two systems, so that if one system crashes, the applications it was running can move to another machine, or node, in the cluster (one feature of MSCS is its ability to restart applications before moving them to functioning machines). From the outside, a cluster appears to be one computer because MSCS supports the concept of virtual servers: MSCS creates a virtual server to represent a particular application. When MSCS moves applications from a failed system to working nodes in a cluster, clients aren't aware of the change because they are talking to the virtual server, which moves with the application—clients do not talk to the node that the virtual server is mapped to. As applications migrate, a client might notice only a pause in service. The virtual server model insures that applications running on a cluster are highly available—only multiple failures would cause a real disruption.

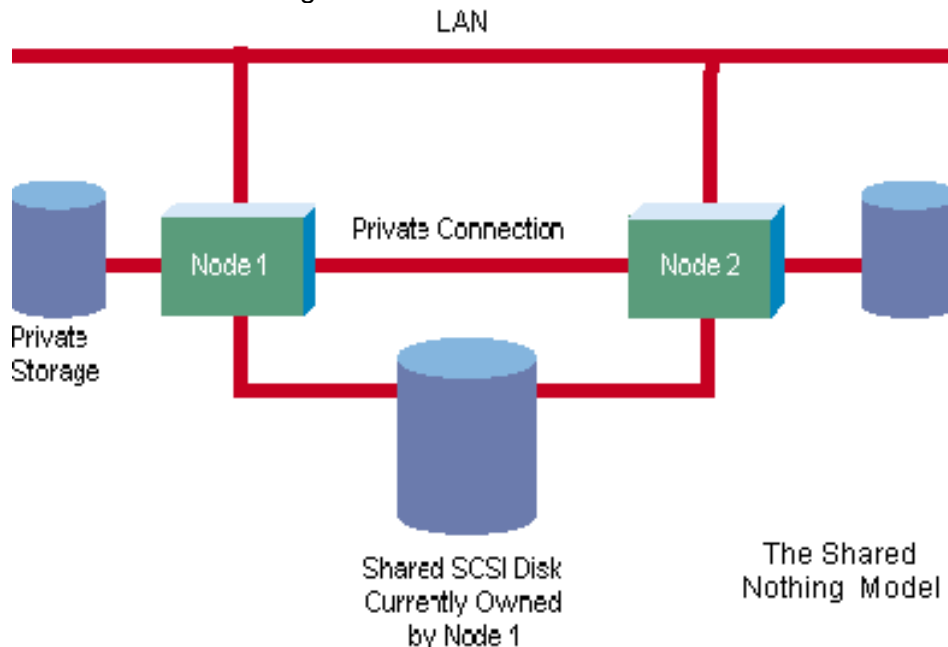
Right now there are two software models employed in clustering that affect the way nodes share hardware: the shared-disk model and the shared-nothing model. In the shared-disk approach, software running on any of a cluster's nodes can access any disk connected to any node, as Figure 1, page 58, shows. Locking protocols coordinate the data consistency the nodes see when they access a disk. The shared-disk model can reduce the need for peripherals and allows for easy data sharing. However, the shared-disk model has a primary disadvantage: As the number of nodes in a cluster grows, so does the locking protocol overhead for accessing shared disks—a situation that can take a toll on performance.

# Inside Microsoft Cluster Server

Mark Russinovich  
(Reprinted from WindowsItPro Magazine)



The shared-nothing model, as Figure 2, page 58, illustrates, assumes that each node in a cluster owns certain disks and that no direct sharing of disks between nodes occurs. Even when disks are connected to multiple nodes, as MSCS requires, only one node can own each disk. A node cannot access a disk it does not own unless the node that owns the disk fails or gives up control of the disk. The shared-nothing model can increase costs because it requires more resources, but it improves scalability because there is no sharing to create bottlenecks.



Clustering contrasts with other methods of making applications highly available. In clustering, there can be a significant pause when a functioning node takes over running applications from a failed node. The length of the pause is application-dependent and can be as short as 30 seconds or as long as 10

# Inside Microsoft Cluster Server

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

minutes. Thus, clustering is a good choice for environments such as Web serving or corporate databases, in which pauses in service are not system-threatening. However, for systems in which an outage of even a few seconds can cause serious problems (e.g., avionics or airline reservations computers), another solution is appropriate. In this setup, applications run simultaneously on two or more computers. The system detects failures when it compares the output from different application instances. If different instances have different output, the system votes or arbitrarily chooses one instance as correct. Then, even when one computer fails, there is no pause in functioning. The multiple-computer approach is more costly than clustering because it requires two or more computers to do the work of one, and it might also require proprietary hardware.

## MSCS Concepts

MSCS clusters consist of nodes, individual computers complete with their own processor, memory, and system disks. Nodes in an MSCS cluster must have access to at least one shared disk. In the initial release of MSCS, this shared disk must be a SCSI disk that is either dual-ported or accessible through two different SCSI adapters attached to the same SCSI bus. The nodes must be connected on a private network separate from LAN traffic. The data files, IP addresses, network shares, and other parts of the installed server applications on the nodes are the cluster's resources. A resource can be active on only one node at a time, and when the cluster detects that a resource has failed, it relocates the failed resource to a different node.

MSCS organizes resources into relational groups. One type of relationship is a dependency relationship. For example, if an application requires a network name and TCP/IP address to become active before the network share comes online, you can specify the TCP/IP address resource, network name resource, and file share resource as resources that belong to the same group (the file share group). When MSCS detects a resource failure, it moves all the resources in the failed resource's group to a different node and restarts the failed resource. In addition, you can establish dependencies between various resources in the same group so that they come online in a specific order.

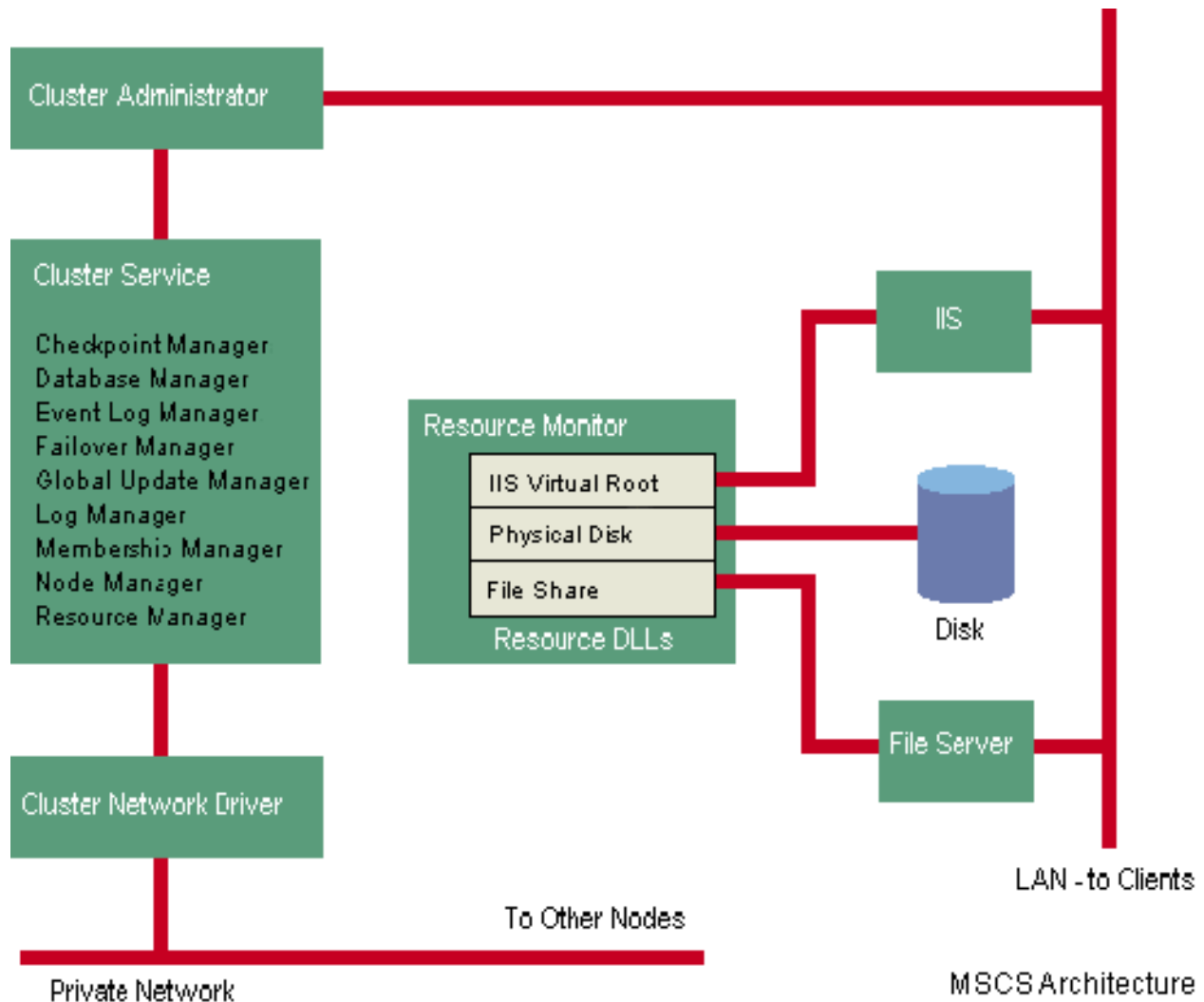
## MSCS Architecture

Although the current release of MSCS implements clusters consisting of two nodes, future releases will support clusters made up of multiple nodes. So with an eye on the future, in the rest of this discussion I'll refer to a cluster as consisting of two or more nodes.

The software components that make up a node in an MSCS cluster are Cluster Service, Cluster Network Driver, Cluster Administrator, one or more Resource Monitors, and one or more Resource DLLs. Figure 3 shows how these pieces relate to one another. Cluster Service, which MSCS implements as an NT service, is the command center of the cluster. Cluster Service is in charge of managing the resources of the cluster and communicating with Cluster Services on the other nodes to coordinate resource ownership and monitor for failures. Cluster Service is made up of several discrete components that divide resource ownership coordination and failure monitoring into subtasks. For example, Cluster Service's Database Manager uses Cluster Service's Log Manager to log transactions.

# Inside Microsoft Cluster Server

Mark Russinovich  
(Reprinted from WindowsItPro Magazine)



Cluster Service components include Checkpoint Manager, Database Manager, Event Log Manager, Failover Manager, Global Update Manager, Log Manager, Membership Manager, Node Manager, and Resource Manager. Checkpoint Manager, Database Manager, and Event Log Manager maintain a coherent image of the central cluster database, which must be stored on the cluster's quorum resource—in the current version of MSCS this resource is a shared SCSI drive. Whenever a change occurs in the status of a node, resource, or group, MSCS updates the cluster's database transactionally. Global Update Manager notifies other nodes in the cluster whenever a change occurs in the status of one node. Maintaining internal node and network configuration information is the job of Node Manager. Node Manager and Membership Manager work together to monitor and control nodes both as active participants in the cluster and as offline nodes.

As in standard NT services, errors that occur in any part of the cluster software are logged to the System event log; however, Cluster Service's Event Log Manager, instead of the native NT event log manager, records these errors. Event Log Manager replicates the cluster event log across the cluster's nodes, ensuring that any node can use its Event Viewer to view cluster errors. Failover Manager handles the task of failing over (i.e., moving cluster resources from one node to another) resources and groups while honoring their start order and dependencies. Resource Manager initiates the migration of resources between nodes by communicating with Failover Manager.

# Inside Microsoft Cluster Server

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

Cluster Network Driver is the intranode communications channel, and it is responsible for providing reliable communications. Cluster Services on the MSCS nodes use the Cluster Network Driver to check other nodes periodically to determine that they are operational (the cluster network manages communications with User Datagram Protocol—UDP). Cluster Network Driver notifies each node's Cluster Service when it detects a communications failure. Then, each Cluster Service determines whether its node should become the only active node in the cluster or the node should fail itself (I will elaborate on this function shortly).

The private network links between nodes in a cluster facilitate intranode communications. Because the links between nodes are crucial to the stability of each node, other traffic should not share these links. However, if you don't install the private connections, or if they fail, the nodes will attempt to use their LAN connections to talk to one another. Communications between applications on a cluster and clients take place over the LAN using a remote procedure call (RPC) on top of the TCP/IP protocol.

MSCS uses resources to abstract the parts of an application that move from one node to another. Examples of resources include network shares, IP addresses, applications, physical disks, and Web server virtual roots. Every MSCS resource requires a Resource DLL, which monitors and controls resources of a particular type. MSCS comes with the handful of standard Resource DLLs shown in Table 1, but most MSCS applications can define their own Resource DLLs.

**TABLE 1: Resource Types Cluster Service Implements**

Resource Type (as listed in Cluster Administrator)	Description
Fault Tolerant Disk Set	Fault-tolerant disk sets, such as striped sets
File Share	File shares accessible by a network path, such as \\Servername\Sharename
Generic Application	Network or desktop applications, such as a database program
Generic Service	Windows NT services, such as a logon authentication service
IIS Virtual Root	Microsoft Internet Information Server (IIS) virtual roots for WWW, FTP, and Gopher
IP Address	Address in Internet Protocol format
Network Name	Friendly name for network device or service
Physical Disk	Disk resources on the shared SCSI bus for shared folders or storage
Print Spooler	Printer queues providing access to a network printer connected to the network by IP address rather than by an individual name
Time Service	Special resource that maintains time consistency between cluster nodes

Resource DLLs load into the memory space of Resource Manager, which executes separately from Cluster Service. The separate execution of Resource Manager insulates Cluster Service from Resource DLLs that fail, and introduces a component that can detect failures in Cluster Service. When Cluster Service queries the status of a resource or starts or stops a resource, it sends the command through Resource Monitor, which contains in its memory space the resource's Resource

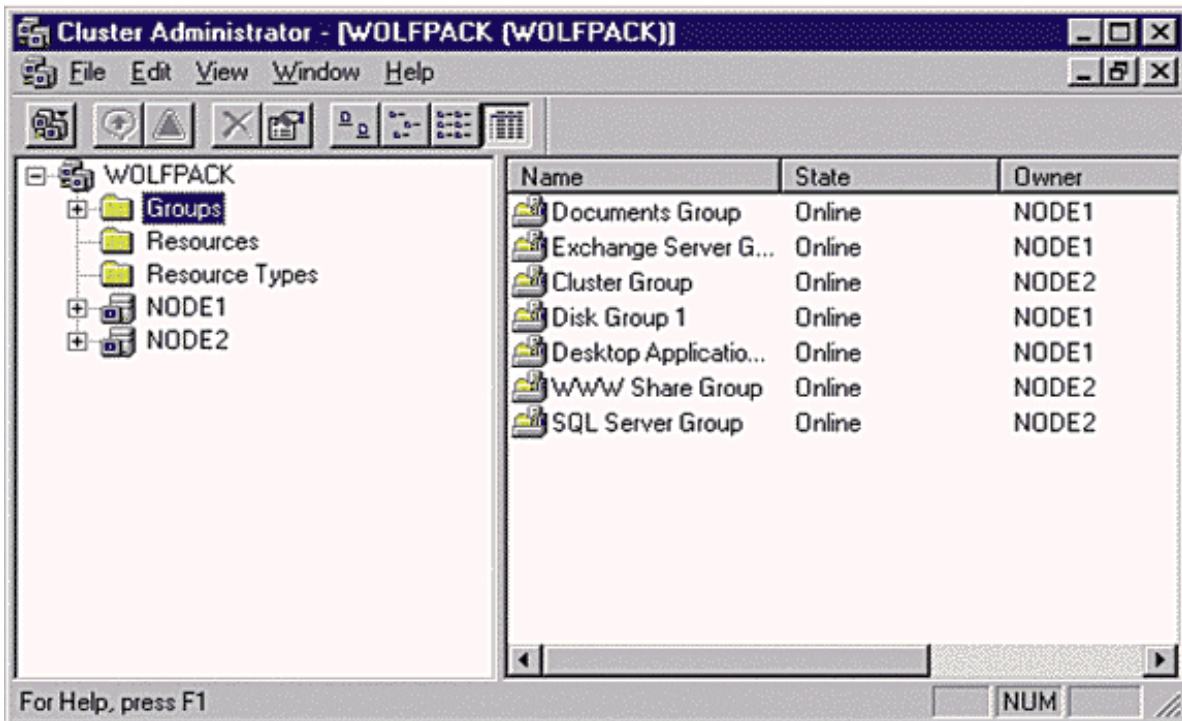
# Inside Microsoft Cluster Server

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

DLL. Cluster Service creates one Resource Monitor by default, but you can create more Resource Monitors manually through Cluster Service. You might want to create extra Resource Monitors if you have a Resource DLL that crashes consistently and you want to separate it from other Resource DLLs.

You use the Cluster Administrator program to configure and monitor clusters, as Screen 1, page 62, shows. Cluster Service exports an administrative API that Cluster Administrator uses to group cluster resources, list resources that are on the cluster or are active on a node, and take resources offline and restart them. Developers can create other cluster administrative tools that make use of this published administrative API.



## Resource States and Failover

Each resource exists in one of five states: offline, offline pending, online, online pending, and failed. When a resource is online, it is active and available for use. Offline pending and online pending are transitional, internal states that reflect a change from online to offline and vice versa. When MSCS notes a resource's state as failed, a failover ordinarily occurs; however, you can effect a manual failover from Cluster Administrator. One type of failure detection that is resource-independent occurs when a node or the communications links between nodes fails. Cluster Network Driver detects these problems and notifies Failover Manager and Membership Manager. The affected nodes determine, with the aid of Membership Manager and Resource Manager, whether they should attempt the failover of their resources.

Individual resources can fail, and the detection of individual resource failure is the domain of each resource's Resource DLL and the Resource Monitor. Resource Monitor pings each resource's Resource DLL at specified intervals during the configuration of the resource to determine if the resource is functional. Resource DLLs implement two entry points for failure detection: LooksAlive and IsAlive. At a configured polling interval, Resource Monitor sends the Resource DLL a LooksAlive request. Upon this request, the Resource DLL performs a cursory check to see if the resource is OK,

# Inside Microsoft Cluster Server

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

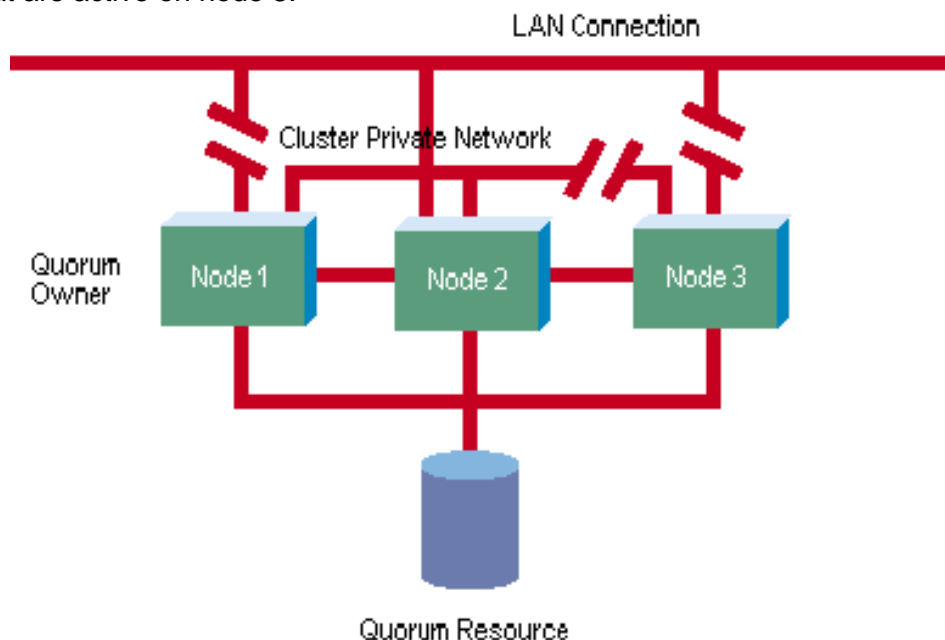
and the Resource DLL must respond to Resource Monitor within 150 milliseconds (ms). If the LooksAlive request fails, Resource Monitor will send an IsAlive request to the Resource DLL. To respond to the IsAlive request, the Resource DLL has up to 300ms to determine if the resource has failed. If the Resource DLL answers no to Resource Monitor in response to the IsAlive request, Resource Monitor considers the resource to be offline and signals to Cluster Service to fail over the group that contains the failed resource.

Resource Monitor has timeout and retry mechanisms that it uses in its detection of failed resources. If Resource Monitor sends an IsAlive request and the queried Resource DLL does not respond to the request within a timeout period, Resource Monitor resends the request a certain number of times (you specify this number in Cluster Administrator). Resource Monitor considers a resource to be offline only when the queried Resource DLL does not respond to Resource Monitor's specified number of request retries.

You will run across the term failback in MSCS. Failback is the process of bringing resources back online on a node that has rebooted and restarted after a failure. You can use Cluster Administrator to configure a resource group to run on certain nodes preferentially. Thus, if MSCS has moved a resource to a nonpreferred node and a preferred node subsequently comes online, the resource might be forced to fail back to the preferred node. You might want to designate preferred nodes when you have a cluster containing nodes with different hardware characteristics, such as different processors and memory sizes. You can designate your SQL Server application to execute on the faster node and your IIS Web server to run on the slower one.

## The Quorum Concept

One of the most important aspects of the successful recovery of failed resources is agreement among all the nodes in a cluster about which node will own the shared disk and where resources will migrate in the failed state. Consider the case illustrated by the three-node cluster in Figure 4. If node 3 becomes separated from the other two nodes, nodes 1 and 2 will assume that node 3 has failed, and node 3 will assume that nodes 1 and 2 have failed. Without some kind of arbitration process, node 3 would fail over all the groups that are active on nodes 1 and 2, and nodes 1 and 2 would fail over all the groups that are active on node 3.



# Inside Microsoft Cluster Server

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

To avoid such a situation, MSCS defines a quorum resource, which all nodes in a cluster must have access to, even though only one node can own the quorum resource at a given time. Through MSCS, the cluster hardware enforces the restriction that only one node at a time can own the quorum resource. If the cluster becomes divided, the quorum resource determines which nodes will remain active. For example, if node 1 owns the quorum resource at the time of a communications breakdown when node 3 has failed, the Membership Managers on nodes 1 and 2 will decide that they are the functioning part of the cluster. Node 3 will attempt to take ownership of the quorum resource, but the hardware lock node 1 imposed on the quorum resource will prevent node 3 from doing so. Node 3 will realize that it has become separated from the other nodes (which are still active) in the cluster, and it will release all its resources and halt operation.

In the current release of MSCS, the only defined quorum resource type is a shared physical disk that supports hardware-based lockout (thus the requirement for at least one shared SCSI disk). Other shared disks can store data, but the quorum shared disk is dedicated to its role in cluster management. In addition to serving as the quorum resource, the quorum shared disk stores the cluster change log and database. When the owner of a quorum shared disk fails, the subsequent quorum owner can finish any cluster-related transactions that are outstanding at the time of the failure after reading them from the log on the quorum disk.

## The Future of MSCS

When I discussed failback, I mentioned that an administrator can configure groups to run on preferred nodes. This practice balances the load among the nodes in a cluster to maximize the return from the hardware—you don't want all your groups to run on one node while other nodes remain idle. Unfortunately, the current release of MSCS lets you only statically distribute workloads between the cluster's two nodes. To address this deficiency, Microsoft

has promised it will release a version of MSCS in 1998 that, in addition to supporting clusters of up to four nodes, will support dynamic load balancing. Dynamic load balancing will simplify the management of MSCS clusters and maximize the use of cluster hardware. Although load balancing in a two-node cluster is relatively straightforward, load balancing in clusters of four or more nodes is much more complex.

As an administrator or developer of enterprise-level software, you will become familiar with MSCS. In time, more and more applications will be MSCS cluster-aware out of the box to adhere to the MSCS standard and take advantage of the high-availability infrastructure MSCS provides.