Cover Feature

Windows NT Clustering Service

The Windows NT Clustering Service supports high-availability file servers, databases, and generic applications and services that are essential to daily business operations.

Rod Gamache
Rob Short
Mike Massa
Microsoft Corporation

A cluster is a collection of computer nodes—independent, self-contained computer systems—that work together to provide a more reliable and powerful system than a single node.[1] To be effective, the cluster must be easy to program and no more complex to manage than a single large computer. Clusters have three advantages over single nodes, They can grow larger than the largest single node, they can tolerate node failures and still continue offering service, and they can be built from inexpensive smaller computers.

In general, the goal of a cluster is to distribute a computing load over several systems, without users or system administrators being aware of the independent systems running the services. If any hardware or software component fails, users might lose access briefly or may experience degraded performance, but they will not lose access to the service.

The Microsoft Cluster Service[2] for Windows NT responds to market needs by addressing high-availability file servers, databases, Web servers, electronic mail servers, and generic applications and services. For many organizations, these servers are essential to daily business operations.

The NT clustering service detects and restarts failed hardware or software components or migrates the failed component's functionality to another node if local restart is not possible. This recovery strategy improves the mean- time-to-repair for the system and thereby improves overall availability.

The cluster service also offers a much simpler user and programming interface. System administrators initially define requirements for operating behavior, and the clustering-service software automatically manages policy decisions regarding a cluster's state as well as applications running on a given node.

NT CLUSTER ABSTRACTIONS

The cluster service uses several abstractions—including *resource*, *resource dependencies*, and *resource groups*—to simplify both the cluster service itself and user-visible system administration.

Resource

A resource is the basic unit of management within the cluster: It allows the cluster software to treat objects managed by the cluster identically, even though they may provide totally different functionality. NT implements several resource types for its clustering service: physical hardware, such as SCSI disks; logical items such as IP addresses, disk volumes, and file server shares; and resource types to allow existing applications to take advantage of the cluster service.

Each resource provides an identical set of interfaces to the cluster software, allowing the resource to be monitored and managed. Resources can inform the cluster service of a failure in the function it represents. In that case the cluster services can restart the resource or any other resource that depends on that resource. Resources may, while under the control of the cluster service, migrate to other nodes in the cluster.

Application developers can design their own resource types and control libraries. To do this, the developer provides a resource DLL to extend the capabilities of the cluster software to manage a specific resource.

Resource dependencies

Resources often depend on the availability of other resources. An instance of an SQL database depends on the presence of physical disks that store the database. These dependencies are declared and recorded in a dependency tree that is maintained by the cluster. The *dependency tree* describes the sequence in which the resources need to be brought online and taken offline, and which resources need to migrate together. Dependencies cannot cross resource group boundaries.

Resource groups

A *group* is a collection of resources that must be managed as a single unit. Usually, a group contains all the elements necessary to either run an application or connect client systems to an application's services. Groups allow a systems administrator to combine resources into larger logical units and manage them as

0018-9162/98/$10.00 © 1998 IEEE

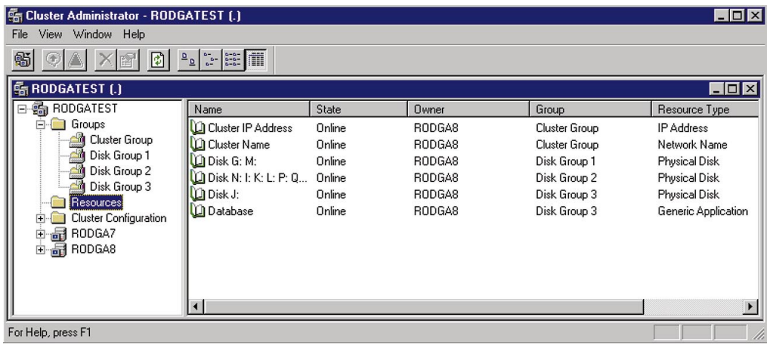

Figure 1. A view of a cluster from the cluster administration tool. The cluster contains two computer systems and defines six resources and four resource groups.

a single entity. Operations performed on a group affect all resources contained within that group.

The group is the unit of failover management, which means that if any resource in the group fails, then the entire group is considered to have failed. For that group to be *failed over* to another system, the new system must be capable of supporting all the resources in the group.

The administrator of the cluster can assign a collection of independent resource dependency trees to a single resource group. When the group needs to migrate to another node in the cluster, all the resources in the group will move to the new location. The administrator sets failover policies on a group basis, including the list of preferred owner nodes, the failback window (the time delay between a failed node rebooting and the cluster service moving applications back onto it), and so forth.

Figure 1 shows a view of a cluster from the cluster administration tool. The cluster itself is called RODGATEST and it contains two computer systems, RODGA7 and RODGA8. Six resources are defined along with four resource groups. Each of these constructs is described in the following sections.

CLUSTER ARCHITECTURE

The key features of cluster operation are resource management, which controls the local state of resources, and failure management, which orchestrates the response to failure conditions.

Windows NT maintains configuration information in a local database, called the *registry*. In clustered systems, a cluster-wide registry allows the cluster service and applications to create and maintain a globally consistent view of the state of its resources as well as the current cluster configuration. Current cluster membership is recorded in the registry. Updates to the cluster registry are done using an atomic update protocol.

In a clustered system it is critical that all nodes maintain a consistent view of cluster membership. Membership agreement is achieved through a protocol based on the Tandem Regroup Algorithm described in more detail later in this article.[3]

The cluster service relies heavily on the Windows core infrastructure, including RPC (remote procedure call) mechanisms, name management, network interface management, security, resource controls, and file system. The cluster extends these base services to all nodes in the cluster with the levels of transparency and reliability described next.

Transparency

The cluster—even though it is built of a group of loosely coupled, independent computer systems—presents itself as a single system to clients outside it. Therefore, client applications interact with the cluster as if it were a single high-performance, highly reliable server. Clients are not affected by interaction with the cluster and do not need modifications to take advantage of the cluster service. However, certain clients may need to use transactional semantics when communicating with the cluster to guarantee that no data is lost in the event of a failure. The clients should keep all information relating to the operation until they know that the server has safely stored the results of the operation onto disk.

Similarly, system management tools access and manage the cluster as if it were a single server. Service and system execution information is available in single-image, cluster-wide logs.

Reliability

The cluster service can detect failures of the hardware and software resources it manages. In the case of an application or node failure, the cluster service can restart failed applications on other nodes in the cluster. The administrator can set the restart policy to meet the availability specification for that application, which is part of the cluster configuration. A failure can also cause the control over resources (shared disks, network names, and so on) to migrate to other nodes in the system. Finally, administrators can upgrade hardware and software in a phased manner without interrupting the availability of the services in the cluster.

We explicitly did not address several issues as part of the first phase of the design because of either technical complexity or schedule pressures. We provided no support for fault-tolerant applications (process pair, primary-backup, active replication), no facilities for the nonstop migration of running applications, and no support for the recovery of the shared state between client and server. These and other "traditional" cluster features will be added in phases to future versions of the product.

VIRTUAL SERVERS

Clients in client-server applications generally connect to a physical server to access their service. In Microsoft Cluster Service (MSCS), we created an

abstraction—a *virtual Windows NT server*—that encapsulates the service and all the resources required to provide that service. The virtual server includes a network name and IP address that are visible and accessible from clients, no matter where in the cluster they are instantiated. Several of these virtual servers can run on a single node, much as Web servers now host multiple home pages at different IP addresses.

The application itself also runs within a *virtual server environment*. This environment provides the application with the illusion that it is running on a virtual NT node with virtual services that appear to be consistent with the application, even if the application is restarted on a different node in the cluster. So the virtual server environment provides the application, the administrators, and the clients with the illusion of a single, stable environment.

A virtual server resource group requires a node name resource (offered by NetBios and Domain Name System) and an IP address resource. Together, these present consistent naming for the clients to access the service. Also, the same resource group needs to include all the resources required to execute the server application (disks, databases, and so on).

Server encapsulation

Each virtual server environment provides a name and configuration space that is separated from other virtual servers running at the same node. Each application can do registry access, service control, and IPC (interprocess communication) and RPC and has a consistent view even if it runs on another node in the cluster. This separation ensures that two instances of an application service running on the same node but in separate virtual server environments will not clash as they access configuration data or communicate internally.

Two significant changes were made to the base Windows NT code to create this virtual server environment.

- We changed the system calls that return or use the system name, such as GetComputerName or gethostname, to return the network name associated with the virtual server instead of the name of the node on which the application is running.
- We modified the NT registry, which stores the application configurations in a single monolithic hierarchical structure, into separate trees that store parts of the registry. Separate trees ensure that applications running in separate virtual servers on the same node can access unique configuration data and store their current state in the registry. Each unique tree represents a single virtual server and is internally dependent on the name associated with the virtual server.

APPLICATION AND SERVICE AVAILABILITY

Applications generally need not be modified to operate within the clustering service environment. There are two modes that a given application can choose to become part of the cluster environment. In the simplest case, any Windows NT application or service can register as a *generic* application or service and use the NT infrastructure supplied. Alternatively, the application can provide a customized resource DLL (dynamically linked library) tailored to work specifically with the given application or service.

Applications generally need not be modified to operate within the Microsoft Cluster Service environment.

Generic application or service

As the name suggests, a generic application or service is one that has not been modified in any way to take advantage of MSCS's features. After having been registered, the application is simply represented by a resource and managed by the cluster service.

However, generic resources have very primitive failure detection and are therefore less reliable than a custom application or service resource DLL. For example, the generic application failure detector simply checks to see if the process has exited to determine if the application is still running. The generic service similarly queries the Windows NT Service Control Manager to see if the service is still running. A hung application or service will not always be detected. For this reason, developers can provide a custom resource DLL for more accurate health detection.

Custom application or service

A custom application or service does not require that the actual application or service be modified. Instead, customization entails the development of a cluster interface layer that allows the cluster software to correctly manage and respond to failures detected by the custom resource DLL. The principal interfaces allow the resource to be started and stopped as well as monitored to see if it is still operating correctly. The advantage of developing a custom resource type is that the cluster software correctly controls the application or service, and a routine can be written to verify that the application is working properly. For example, a database service failure detector could send database queries into the service to detect if the service is operational. This level of sophisticated detection cannot be done with a generic service, since it knows nothing about the specific service.

NETWORK INFRASTRUCTURE

Modern applications such as client-server, distributed, and management applications all rely on networks in one form or another. The capability to detect and survive network failures is essential for mission-

The clustering service uses the COTS model and improves network reliability by using all available networks, whether private or public.

critical, line-of-business applications and systems.

The Windows NT Clustering Service improves the availability of Windows NT networks by

- continuously monitoring the state and quality of all networks used by the clustering service;
- transparently and reliably switching the traffic among networks if multiple networks are available; and
- automatically failing over network resources among cluster nodes.

To explain the operation of the networking software, we first introduce a high-level overview of the operation of the cluster service as a whole.

Clustering service network components

Figure 2 shows an overview of the components and their relationships in a single system of a Windows NT cluster. The clustering service controls all aspects of cluster operation on a cluster system. It is implemented as a Windows NT service and consists of eight closely related, cooperating components:

- *Node manager* handles cluster membership and watches the health of other cluster systems.
- *Configuration database manager* maintains the cluster configuration database.
- *Resource manager/failover manager* makes all resource and group management decisions and initiates appropriate actions, such as startup, restart, and failover.
- *Event processor* connects all components of the cluster service, handles common operations, and controls cluster service initialization.
- *Communication manager* manages communications with all other nodes of the cluster.
- *Global update manager* provides a global update service that is used by other components within the cluster service.
- *Resource monitor* monitors the health of each resource via callbacks to the resources. Strictly speaking, it is not part of the cluster service. It runs in a separate process, or processes, and communicates with the cluster service via RPC.
- *Time service* maintains consistent time within the cluster but is implemented as a resource rather than as part of the cluster service itself.

The node manager, resource manager (with its resource monitors), and communication manager are particularly interesting in the context of this article.

Node manager. Node manager maintains cluster membership and sends periodic messages, called *heartbeats*, to the other systems in the cluster to detect system failures. If one system detects a communication failure with another cluster node, it broadcasts a message to the entire cluster, causing all members to verify their view of the current cluster membership. This is called a *regroup event*. If this happens, the writes to potentially shared devices are frozen until the membership has stabilized. If a node manager on a system does not respond during the regroup, it is removed from the cluster and its active groups must be *failed over* ("pulled") to an active system. Note that the failure of a clustering service also causes all of its locally managed resources to fail.

Resource manager. Resource manager/failover manager is responsible for stopping and starting resources, managing resource dependencies, and initiating the failover of groups. It receives resource and system state information from the resource monitor and the node manager. It uses this information to make decisions about groups. Resource monitor monitors the health of a resource by periodically calling the IsAlive and LooksAlive callback functions in the resource DLLs.

Communication manager. Communication manager is responsible for providing communication among all nodes. In conjunction with a kernel mode transport driver, it provides a reliable messaging infrastructure to the rest of the cluster software. The kernel mode driver uses all available networks and, when one fails, reliably and transparently switches the traffic to the next available network.

Network reliability

The reliability of network hardware components has dramatically improved over the past few years. However, failures still happen. For mission-critical applications, the reliability offered by standard hardware is not sufficient. There are two alternatives:

- Use expensive, fault-tolerant, proprietary products.
- Use COTS (commercial-off-the-shelf) components that provide similar reliability via software.

The clustering service uses the COTS model and improves network reliability by using all available networks, whether private (node to node) or public (cluster to client). When one network fails, the cluster communication traffic is transparently moved to the next available network.

Availability. Most services provided by a clustered server are accessed through either of two resource types: a network name or an IP address. A *network name* is a friendly name of a device or service; it is published by the clustering service and included in any

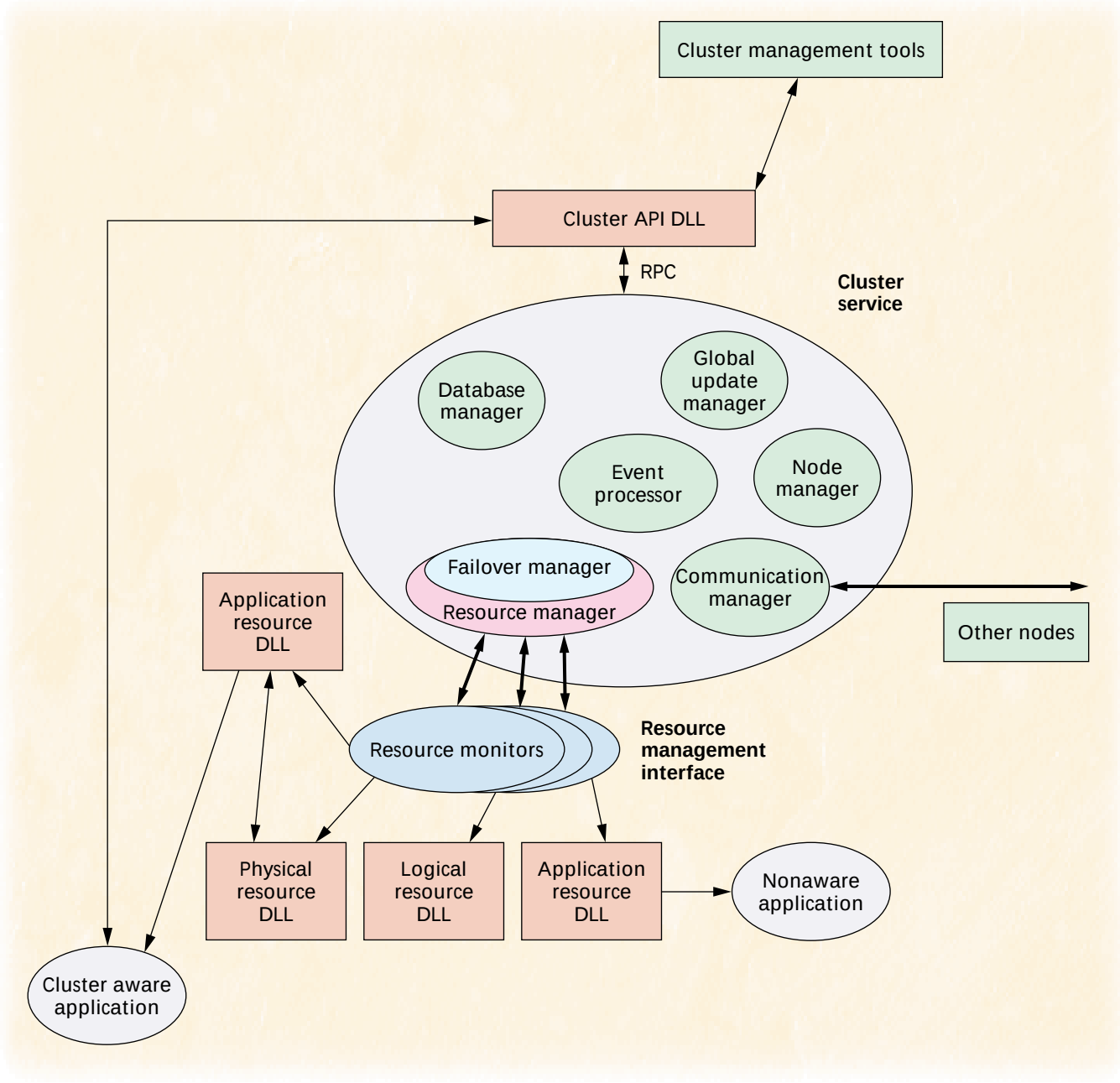

Figure 2. Cluster service network components. The clustering service controls all aspects of cluster operation on a cluster system. It is implemented as a Windows NT service and consists of eight closely related, cooperating components.

available naming services, such as DNS and Windows Internet Naming Service (WINS). An *IP address* is a 32-bit number in dotted decimal format that represents an Internet Protocol (IP) address.

The availability of cluster resources can only be as good as the availability of the network used to access the cluster. For such services to be highly available, it is essential that any network failures be properly detected and the function failed over and restarted on another node or network.

Here we describe how the clustering service monitors the health of network components, what mechanisms are used to detect network component failures, and the actions that are taken in case such failure is detected.

Failure detection. Heartbeats are the primary method of detecting communication failures and determining their scope. A heartbeat is a simple message sent to another node in the cluster approximately every second. If more than two heartbeat messages are not received in a period of time, the cluster service decides that the network has failed. If a further three heartbeat messages are not received, the cluster service initiates a regroup event and makes every attempt to determine the state of the other node. The hardware interface driver may be queried to determine if low-level hardware failures have occurred.

The two-node case is especially problematic, because of the potential damage done by a node incorrectly deciding that the other node is down. To help in fault isolation, a secondary method of pinging external hosts or routers to see if they can communicate with the missing node is tried. A quorum algorithm

Changes in the states of the networks and network interfaces in a cluster must be consistently, accurately, and swiftly reported so that administrators and application software can take appropriate actions.

is then used in conjunction with other fault isolation mechanisms to determine if either node can continue to operate as the surviving cluster member.

Network resource failover process. Based on the results of the network state analysis, the clustering service can determine if it should failover network resources. Three assumptions were made during the design process:

- Clients must use IP protocol, as the clustering service can failover only IP addresses.
- Clients must know how to gracefully reconnect or retry after failure.
- All cluster nodes must be on the same LAN segment.

During the failover process, the clustering service moves the network name and IP address to the surviving node. It rebinds the IP address to the new MAC (media access control) address and transmits an ARP (address resolution protocol) broadcast containing the new mapping, thus flushing all ARP caches on the local segment.

When an address is instantiated on a surviving node, the transport on that node will begin receiving packets from all clients that were connected to the primary prior to the failover.

Managing network state

Changes in the states of the networks and network interfaces in a cluster must be consistently, accurately, and swiftly reported so that administrators and application software can take appropriate actions. Determining the location and scope of a fault, to the greatest extent possible, is an important part of this process. Such a determination enables applications to work around the problem and helps administrators make repairs.

When a failure occurs that can be isolated to a subset of the nodes on the network, IP address resources that depend on the affected interfaces should fail. The failure of an IP address resource will trigger a failover of the associated resource group to another node, which can continue to offer service.

Network and interface states. Several rules must be followed when determining the network and interface states:

- All nodes must have a consistent and accurate view of interface and network states across the cluster.
- It must be possible to determine whether faults affect the entire network or only a subset of the attached nodes.
- Outages that are not long enough for other nodes to assume that a node has failed are not reported outside of the network software.

Only a node-down event is reported when a node suddenly crashes. Reporting all possible intermediate state transitions would slow the recovery process in the normal case, while providing no benefit in unusual failure scenarios.

Interface and network state definitions. Before describing the algorithms for network state and failure detection, we define and categorize the pertinent network management objects.

Two objects are of particular interest and are treated separately by the network software.

- The interface includes the hardware network interface and its low-level driver software.
- The network is the communication link between two or more interfaces.

This distinction is important since an interface may fail, but the network connection to another node still works over a second set of interfaces. Each node tracks the state of each network and interface with which it may need to communicate.

Network management process. Each node maintains a *connectivity vector* for each network to which it is attached. The connectivity vector is indexed by node ID and contains the node's view of the state of all interfaces on the network. Connectivity is determined by heartbeat exchanges. A single node, dubbed the *accountant*, performs all state calculations.

Vector entries are calculated using network topology information, node operational status information, heartbeat status information, and driver hard-error notifications. The vectors from all nodes are combined to form a complete connectivity matrix for each network. The matrix is used to calculate the state of the network and the state of each constituent interface. Network and interface states are recalculated whenever a network or interface state transition is reported anywhere in the cluster.

The node that forms a cluster automatically assumes the role of accountant. Other nodes are informed of the identity of the accountant when they join the cluster. Whenever a node detects a change in its connectivity with other nodes, it sends a message containing an updated connectivity vector to the accountant. As connectivity reports arrive, the accountant recalculates the network and interface states. The accountant broadcasts the resulting interface and network state changes to all nodes using a global update.

If the accountant dies, the surviving node with the smallest node ID automatically assumes the role of accountant. The new accountant issues a global update announcing its new duty and directing all nodes to report their current connectivity vectors. The new accountant then recalculates the network and interface states and broadcasts the results.

USING THE CLUSTERING SERVICE

A number of major server applications take advantage of the functionality offered by MSCS. We describe three of these servers: Microsoft SQL server, Oracle Parallel and Fail-Safe servers, and the SAP R/3 business application development system.

Microsoft SQL Server

SQL Server 6.5 is a client-server relational database system that consists of a main server process and a helper process, called the *executive*.

The SQL server process manages a collection of databases that in turn map down to the NT file system. The server is a *free-threaded process* that listens for SQL requests and executes the requests against the database. It is common to encapsulate the database with stored procedures that act on the database. Client requests invoke these procedures, written in the TransactSQL programming language, and the procedures execute a program that has control flow and can make read and write calls to the database. These procedures can also access other NT services (for example, mail is often used for operator notification).

The SQL executive performs housekeeping tasks, such as launching periodic jobs to back up or replicate the database so that it can be used in case the primary system fails.

SQL Server database failover. Using MSCS it was possible to design SQL Server to use server failover as its mechanism for high-availability. We configured the SQL server resource group as a virtual NT server with a virtual registry, virtual devices, virtual services, virtual name, virtual IP address, and whatever else was needed to create the fiction SQL Server was a virtual NT node. Clients connect to the server that runs on this virtual node. The server application thinks it is running on this virtual server, and the virtual server can migrate among physical servers.

This design allows a two-node MSCS cluster to have two or more high-availability SQL Servers. The clients always see the same server name, even as the server migrates from node to node. The server sees the same environment, even as it migrates as well. Administrators, who are really just clients, administer virtual servers, just like real servers (no new concepts have been added). This simple design has proved to be very easy to understand and easy to explain to users.

Oracle database servers

Oracle has developed the Oracle Fail-Safe database server for the MSCS market. This product uses the MSCS *shared-nothing* model. It is based on a data-partitioning pattern in which each instance of the Fail-Safe database server manages one or more databases. Each database is stored at a shared disk but—in conformance with the MSCS model—this disk is accessible only to the node that *controls* it as a cluster resource.

Each Fail-Safe server is a dedicated server, running at each node in the cluster. It maintains the correct dependencies among resources, databases, and database instances. It also monitors the database (by interacting with the resource monitors) and maintains the dynamic configuration of the client-server interface modules.

Clients access a database through the network names and IP addresses associated with the resource group that holds each database, respecting the virtual server model. If a database fails, the client briefly pauses and then reconnects to the database server under the same name and address, which will have moved to the surviving node. Application builders are encouraged to maintain transaction state information, such that after a failure any in-flight transactions can be quickly resubmitted.

MSCS manages the collection of resources that make up each instance of a Fail-Safe database server. In case of failure, the MSCS resource group (disk, database, IP address, and network name) is migrated to another node in the cluster. Once the resources are brought online, the local Fail-Safe server is notified of the new resources and the server instantiates new database servers for each of the new databases.

A number of major server applications take advantage of the functionality offered by Microsoft Cluster Service: Microsoft SQL Server, Oracle Parallel and Fail-Safe servers, and SAP R/3.

SAP R/3

SAP AG offers a scalable system for the development of business applications. SAP builds its systems around a sophisticated middleware system, SAP R/3. R/3 is a three-tier client-server system. Its three tiers separate presentation, application, and data storage into clearly isolated layers. Although all the business logic is in the application tier, there is no persistent state at the application servers. This separation allows application servers to be added to the system to achieve higher processing capacity and availability. Load balancing and availability management of the parallel application servers is done with dedicated R/3 management tools.

All persistent state of R/3 is maintained in the database tier, where a third-party database is used for data storage. To facilitate the use of different database vendors, the system avoids the storage of nonportable elements such as database stored procedures. There are three components, of which only a single instance can be active in the system and which are thus a single point of failure: the database, the message server, and the enqueue server. The failure of any of these servers will bring the complete system to a halt.

MSCS makes these services highly available. Because only a single instance of each server can be active in the system, MSCS can use an *unpartitioned failover*

organization: One node of the cluster hosts the database server and the other provides the message and enqueue services. If either node fails, the server and disk resources are moved to the remaining node in the cluster. The resource group that holds the network names and IP addresses under which the server was accessible also migrates to the surviving node.

The application servers in SAP R/3 are *failover aware* in that they can handle being disconnected from the database or the message/enqueue server temporarily and after a waiting period will try to reconnect again. A failure in the database tier is transparent to the user, as the application servers will mask the potential transaction failure that was the result of the failure. Migration of the application server is handled at the client by establishing a new login-session at the new node hosting the application server.

Microsoft Cluster Service has been shipping for about a year on Windows NT version 4.0. The upcoming Windows NT 5.0 release of Windows NT Clustering Service will improve ease of use through a wizard that guides the user through the creation of cluster resources. Windows NT 5.0 Clustering Service will also improve the integration with existing services. For example, it will provide a more available and easily managed DHCP (dynamic host connection protocol) service.

In the future we will provide the infrastructure for building robust, distributed, and scalable enterprise-level applications. We are adding support for System Area Networks with high-bandwidth, low-latency interconnects. We will provide the system-level infrastructure for building multinode clusters, as well as a complete application environment based on the forthcoming COM+ version of Microsoft's object model technology. ❖

Acknowledgments

We thank Werner Vogels at Cornell and Jim Gray at Microsoft Research for sharing their knowledge and expertise. Tom Phillips and Bohdan Raciborksi provided some badly needed help, since writing this article overlapped with the beta 2 release of NT5.

References

1. G.F. Pfister, *In Search of Clusters: The Coming Battle in Lowly Parallel Computing*, Prentice Hall, Englewood Cliffs, N.J., 1995.
2. W. Vogels et al., "The Design and Architecture of the Microsoft Cluster Service—A Practical Approach to High-Availability and Scalability," *Proc. 28th Symp. Fault-Tolerant Computing*, CS Press, 1998, pp. 422-431.
3. R. Carr, "Tandem Global Update Protocol," *Tandem Systems Rev.*, Vol. 1, No. 2, 1985.

Rod Gamache is responsible for leading the development of the NT cluster product. He has been at Microsoft for the past four years working on networking and Windows NT clusters. Previously, Gamache was at Digital Equipment Corp. for 17 years, where he worked on porting Windows NT to Alpha, the VMS operating system, and networking.

Rob Short is director for development of core operating system infrastructure for Windows NT. Apart from a year working on interactive television, he has worked on Windows NT since starting at Microsoft in 1988. Before that he spent 15 years at Digital designing hardware and software. Short recieved an MS in computer science from the University of Washington.

Mike Massa is responsible for the networking components of Microsoft clusters. He has been a member of the Windows NT development team at Microsoft since 1991. His areas of focus are network protocols and distributed systems. Massa received a BSE in computer science from the University of Pennsylvania.

Contact the authors at Microsoft Corp., One Microsoft Way, Redmond, WA 98052; rodga@microsoft.com.