Rate this page: 10 users 4.9 out of 5

# Building a Highly Available and Scalable Web Farm

## Duwamish Online

Paul Johns and Aaron Ching
Microsoft Developer Network

December 2000

**Summary:** This article discusses how to set up a highly available and scalable Web farm with Network Load Balancing (NLB), using the load-balancing solution implemented by Duwamish Online as an example. Three methods of load balancing are highlighted: Round Robin Domain Name System (RRDNS), load-balancing switches, and Microsoft Windows 2000 NLB. (13 printed pages)

**Contents**

Introduction
Load-Balancing Solutions for Web Servers
Duwamish Online Web Farm with NLB
Conclusion
Appendix: How Round Robin DNS Works

## Introduction

An organization's Web applications often perform mission-critical tasks, where the cost of unavailability can quickly exceed the cost of the application and infrastructure. It's important that these applications be highly available and reliable. An easy and effective way to achieve availability and reliability is to use redundant servers. If you have more than one server implementing your site, and then one of the servers crashes, the processing requests can be redirected to another server. This provides a highly available Web site.

Adding just a few servers can increase your availability significantly. If you have one server with only 90-percent availability, that would mean it's not available for an average of 2.4 hours a day. (Most servers are far more reliable than this, but this extreme example illustrates our point.) The probability of the server failing at a given moment is 0.1. Adding just one additional, similarly unreliable server decreases the probability that both servers will fail at once to 0.1 * 0.1, or 0.01. The likelihood of one of the servers being available is increased to a much-improved 99 percent, or only 14.4 minutes unavailability each day. With seven unreliable servers, the probability of at least one server being available increases to 99.99999 percent. The availability will be far greater with better hardware, of course. Adding redundant servers is an especially appealing and cost-effective strategy for increasing availability when you consider the relatively low cost of Microsoft® Windows® 2000 hardware.

Adding additional Web servers also helps your site handle larger loads in an elegant manner: Adding Web servers can provide near-linear scalability (meaning that throughput increases linearly with additional servers) when guidelines for good cluster design, covered in this article, are followed.

Figure 1 shows the performance of the Duwamish Online site, with the addition of more Web servers.

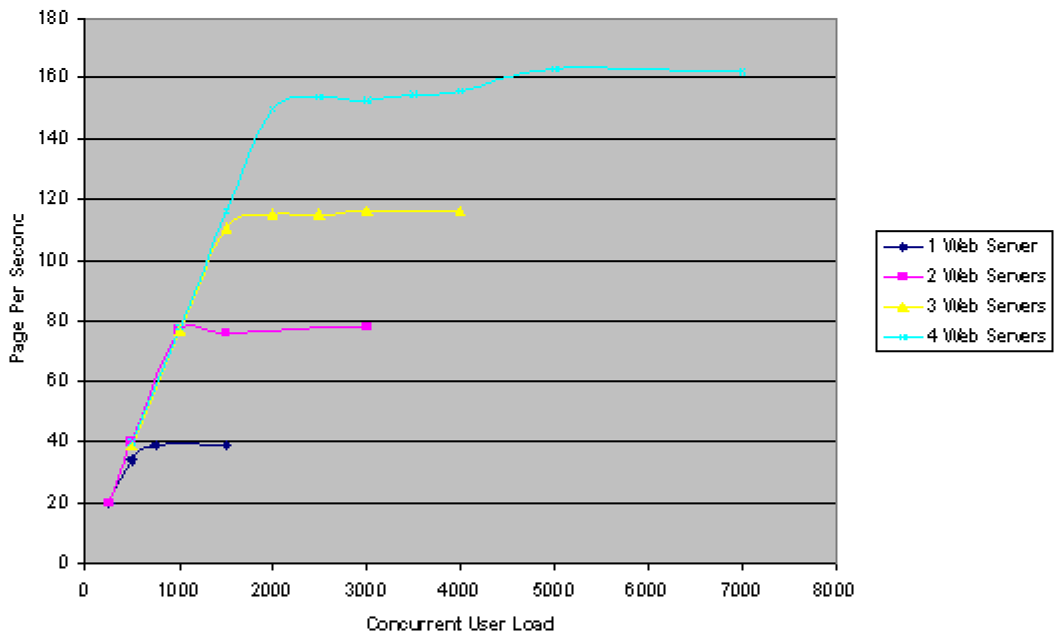

**Figure 1. Duwamish Online scalability as more Web servers are added to the Web farm**

When you implement a site on a Web farm, the processing load must be distributed across the hosts. The processing requests received by the cluster are directed to the different hosts in the cluster, so that response time is low and the application throughput is high. Each host in the cluster runs a separate copy of the server programs required by the application. For example, each host might be a Web server, a File Transfer Protocol (FTP) server, or an e-mail server. Load balancing distributes the incoming client requests across the cluster hosts.

## Load-Balancing Solutions for Web Servers

There are a number of ways of implementing load balancing. For the purpose of our discussion, we will limit ourselves to the three primary methods of load balancing:

- **Round Robin Domain Name System** (RRDNS)
- **Load-balancing switches**, such as Cisco LocalDirector, F5 Networks' BIG-IP, and Alteon Websystems ACEdirector
- **Windows 2000 Network Load Balancing**

We will now look at each of these load-balancing solutions in more detail.

### Round Robin DNS

RRDNS is a simple and low-cost solution for enabling a limited form of Transmission Control Protocol/Internet Protocol (TCP/IP) load balancing for Internet server farms. It usually comes free of charge as a standard feature of most popular operating systems, such as Microsoft Windows NT® 4.0 with Service Pack 4 and Windows 2000, as well as many other systems with support of BIND 4.9 and later.

**Note** Berkeley Internet Name Domain (BIND) is the de facto reference implementation of DNS.

RRDNS allows a pool of servers to appear as a single host to the clients, when in reality client requests are directed alternately to all servers in the pool and the traffic, therefore, is distributed across the servers.

Although Round Robin DNS is popular among many cost-sensitive sites, it does not function effectively as a high-availability solution. In the event of a server failure, RRDNS continues to route requests to the failed server until it is manually removed from DNS, and even then many users must wait for DNS to time out their connection before being able to successfully access the target Web site.

For a more in-depth discussion on how RRDNS works, please refer to the Appendix.

## Load-Balancing Switches

Load-balancing switches, such as Cisco LocalDirector, F5 Networks' BIG-IP, and Alteon Websystems ACEdirector redirect TCP/IP requests to multiple servers in a server farm, providing a highly scalable, interoperable solution that is also very reliable.

These switches sit between the connection to the Internet and the Web farm. All requests come to the switch using the same IP address, and then the switch forwards each request to a different Web server based on various algorithms implemented in the switch. Switches will frequently be able to ping the servers in the farm to make sure they're still up, and to get an estimate of how busy they are—so they can be relatively intelligent about load balancing.

Another common algorithm is to load balance based on the content of the request—perhaps the IP address of the requester, or some other information in the request. Using the IP address alone doesn't work well because some Internet service providers (ISPs) and some companies use proxy servers that change the IP address of all of the requestors that go through the proxy to the same address.

Using a load-balancing switch is much better and more scalable than using Round Robin DNS, but switches can be quite expensive—and you'll need multiple switches to avoid making the switch the single point of failure for your entire Web application. The next solution, Windows 2000 Network NLB, is often less expensive than a load-balancing switch and avoids having a single point of failure.

## Windows 2000 NLB

Network Load Balancing (NLB) is one of the clustering technologies available in Windows 2000 Advanced Server and Windows 2000 Datacenter Server. It provides high availability and high scalability to IP-based applications, such as Web servers and Lightweight Directory Access Protocol (LDAP) servers. Microsoft Application Center 2000 provides NLB on Windows 2000 Server (rather than Advanced Server).

NLB distributes incoming IP traffic across a cluster of multiple servers that provide TCP/IP services. It utilizes a common virtual IP address for the entire cluster and transparently partitions client requests across the multiple servers in the cluster.

Figure 2 shows how NLB works.

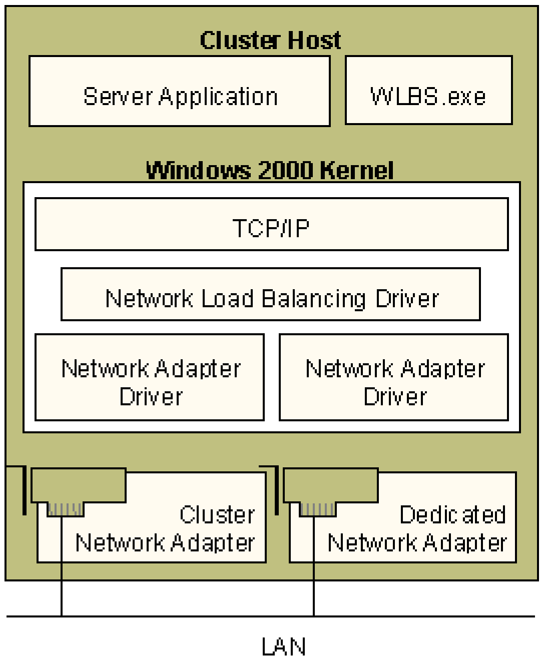

**Figure 2. Network Load Balancing architecture**

In addition to the common external IP address for the cluster, each server in the cluster responds to a dedicated network address as well. So, each machine responds to two network addresses: a *cluster network address* and a *dedicated network address*. Network Load Balancing is implemented using a network driver that is logically placed between the higher-level protocol TCP/IP and the network adapter of the host. All the cluster hosts receive the incoming traffic. The NLB network driver acts as a filter and allows the host to process only a part of the incoming traffic. The incoming requests are accepted according to the NLB settings for the host. We will look at NLB settings later.

The benefits of deploying NLB include:

- **Fault tolerance.** Network Load Balancing automatically detects a nonfunctional host and can recover from it. In the case of an offline host, the incoming traffic is distributed across all the servers. It requires no additional hardware to avoid single points of failure.
- **Higher scalability.** Network Load Balancing supports clusters of up to 32 computers, although smaller clusters of 16-24 computers usually result in better resource utilization. If you need more power, you can load balance across multiple NLB clusters using a combination of NLB and one of the two-load balancing methods discussed previously.
- **Higher performance.** Unlike most load-balancing switches, the performance of Network Load Balancing automatically improves with the speed of the cluster hosts and local area network (LAN), ensuring that it will never become a bottleneck as LAN and host speeds increase.
- **Manageability.** You can specify load balancing for a single TCP port to customize processing load for a host. You can also block access to certain ports.
- **Ease of use.** Network Load Balancing installs a standard networking driver component and does not need any special hardware to run. It's easy to put machines into and remove machines from the cluster—and it will get even easier when the Application Center 2000 server ships later this year. (For more information, see http://www.microsoft.com/applicationcenter/default.htm).

## Duwamish Online Web Farm with NLB

Let's look at the load-balancing solution that has been implemented by Duwamish Online. For Duwamish Online, the network hardware uses the 100 -Mbps switch.

The switch corresponds to the Open Systems Interconnection (OSI) Data Link Layer. It provides dedicated collision-free communication between network devices and enables multiple simultaneous data transmission between two ports, increasing network capacity.

When a switch starts, it builds a table of the media access control addresses of each network adapter card with the port number to which the adapter is connected. Therefore, when a host on Port 1 needs to send a data packet to another host on Port 2, the switch directly forwards the data packet to Port 2. This way, no other host or port needs to respond to data packets in order to be transmitted to Port 2.

In Network Load Balancing, the media access control address is replaced with the media access control address of the cluster network adapter. This media access control address is used by all of the hosts in the cluster. For an incoming request, the switch forwards the request to all of the ports associated with the cluster media access control address. Based on a hashing algorithm, one of the hosts will accept the packet and the other hosts will ignore it. After processing, the data packet is sent back to the port from where the packet was originally received.

We have used the switch because a switch can handle multiple conversations at the same time, at the full bandwidth. Therefore, if one cluster host is processing one request, and the cluster receives another request, another host can pick up the data packet and begin processing. This ensures high system performance.

Figure 3 shows the network architecture of the Duwamish Online site and how the switches are currently deployed.

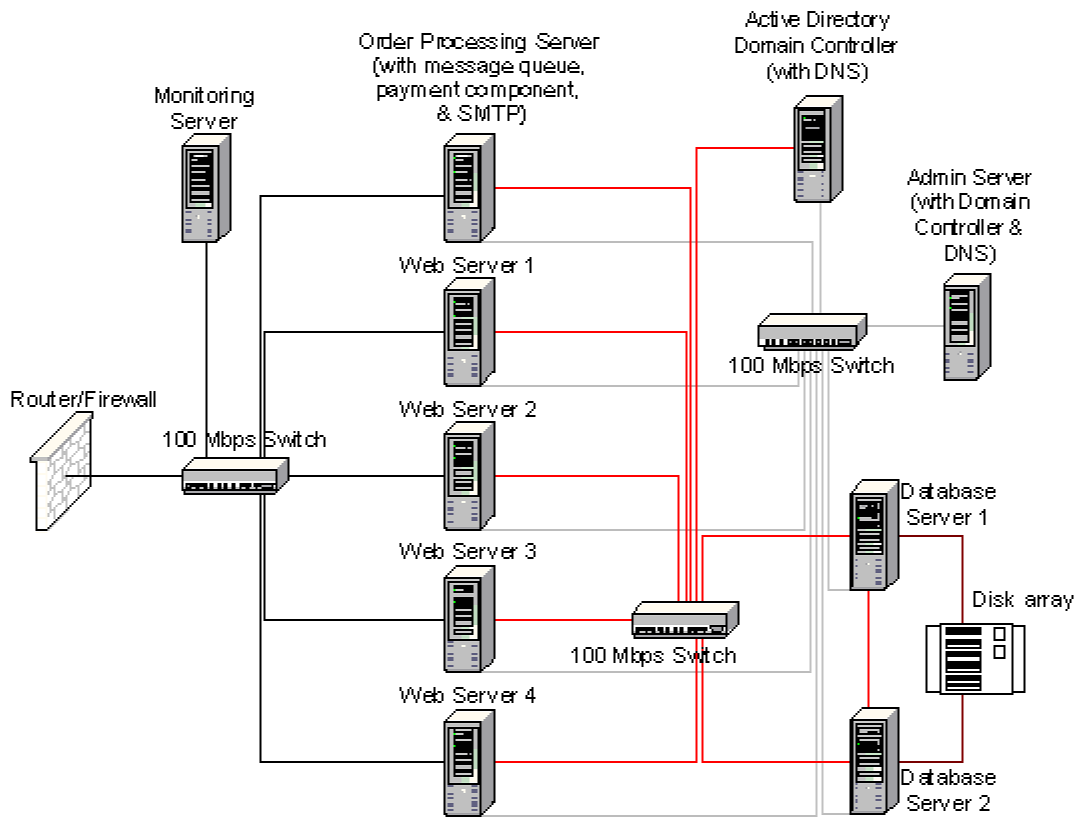

**Figure 3. Duwamish Online hardware configuration**

## Setting Up an NLB Cluster

For each host in a cluster, you need to specify certain NLB settings. Also, you need to specify some parameters for the entire cluster. Therefore, the cluster-level settings need to be identical for all hosts in the cluster.

**To configure NLB for a server**

1. Right-click the **My Network Places** icon on the Windows desktop and select **Properties**. Right-click the network adapter that is connected to the external network segment from the Network and Dial-up Connections window. Select **Properties** to open the **Local Area Connection Properties** dialog box.
2. Select the **Network Load Balancing** check box and click the **Properties** button to open the **Network Load Balancing Properties** dialog box.
3. Click the **Cluster Parameters** tab.

   Cluster parameters must be consistent across all servers in the cluster. Some of the cluster parameters are described in the following table.

   **Table 1. Cluster parameters**

   | Parameter | Description |
   |---|---|
   | Primary IP address | This address is specified in the standard dotted notation and is the virtual IP address of the cluster. |
   | Subnet mask | This parameter specifies the subnet mask of the cluster and is specified in the standard dotted notation. |
   | Full Internet name | This parameter specifies the Internet name for the cluster, as will be accessed by the clients.<br>For the Duwamish Online site, we use "www.duwamishonline.com." |
   | Network address | This parameter specifies the media access control address of the cluster that will be used for handling client-to-cluster traffic. This address is generated based on the cluster's IP address. Duwamish Online uses the default value set by NLB. |

4. Click the **Host Parameters** tab.

   Host parameters are unique for each host in the cluster.

   **Table 2. Host parameters**

   | Parameter | Description |
   |---|---|
   | Priority ID | This parameter specifies the priority for handling traffic for TCP and UDP ports. Each machine must have a unique value. Duwamish Online uses the values 1, 2, 3, and 4 for the four machines in the cluster. |
   | Initial cluster state | This parameter specifies whether Network Load Balancing should start and the host joins the cluster when Windows 2000 is started on the host. Duwamish Online uses the default, which is active. |
   | Dedicated IP address | This parameter specifies the IP address to be used for traffic not associated with the cluster. |
   | Subnet mask | This parameter specifies the subnet mask for the IP address specified. |

5. Click the **Port Rules** tab.

   You need to specify rules that define how each TCP and UDP port cluster traffic is handled.

   Port rules are specified for contiguous address ranges. Port 443 requires special configuration because it is used by Secure Sockets Layer (SSL). Add the port rules listed in the following table.

   **Table 3. Port rules**

   | Port range | Port rules |
   |---|---|
   | 0-442 | Filtering modes: Multiple hosts<br>Special client affinity: None |
   | 443-443 | Filtering modes: Multiple hosts<br>Special client affinity: Single |
   | 444-64000 | Filtering modes: Multiple hosts<br>Special client affinity: None |

You can specify one of the following three filtering modes for the host:

- ***Multiple hosts*** specifies that network traffic be handled by multiple hosts in the cluster.
- ***Single host*** specifies that network traffic associated with a port be handled by a single host in the cluster.
- ***Disabled*** specifies that all network traffic for a particular host is blocked.

For the Duwamish Online site, we selected *multiple hosts* for most ports, so that the traffic is distributed evenly across all hosts in the cluster.

You can also specify client affinity to ensure that a specific host in the cluster handles traffic from a particular client:

- ***None*** indicates that Network Load Balancing does not need to direct multiple requests from the same client to the same host.
- ***Single*** indicates that multiple requests from the same client should be directed to a specific host.
- ***Class C*** indicates that Network Load Balancing directs multiple requests from the same TCP/IP class C range to the same host.

For the Duwamish Online site, we specified *none* for most ports so that requests from the same client can be processed by multiple servers.

### Removing Nodes from a Cluster

It is often necessary to take down a Web server node for scheduled maintenance tasks, such as system hardware or application software upgrades. In other cases, one of the Web servers might fail to respond properly and the operations team might decide to take the node down for further investigation.

It is very simple to remove a Web server node from the NLB cluster:

1. From the command prompt, type in "wlbs stop" and the node will be disjoined from the cluster.
2. From the command prompt of another existing node of the cluster, type in "wlbs query" to verify that the priority ID of the removed node has disappeared.

The operations team can now do whatever tasks they choose to the removed node because it is no longer taking any requests from the Internet.

### Adding Nodes to a Cluster

As traffic to the site increases, you might want to add more Web server nodes to the NLB cluster to increase capacity for the additional load. Here are the steps to expanding an NLB cluster, up to a maximum of 32 nodes in a cluster:

1. Use a server machine with a similar or identical hardware configuration to the other existing Web servers.
2. Install exactly the same software to the new server, including the operating system, any third-party software, and all components of the application.
3. Apply exactly the same system configurations and performance-tuning procedures to the new server.
4. Use the same parameters of NLB settings for the server, with the exception of a unique priority ID and dedicated IP address.
5. Verify all network cabling is correctly installed. Boot up the new server.
6. From the command prompt, type in "wlbs start" and the server will join in as a node of the NLB cluster.
7. From the command prompt, type in "wlbs query" to verify that the priority ID of the new server appears as part of the expanded cluster.

### Managing Application State in a Cluster

In a cluster, it is difficult for the server farm to maintain application state (that is, data specific to a client session). When a single Web server is used, session state may be stored on the Web server. However, load-balancing schemes are most effective when no affinity is maintained between a client and server so that requests from the same client are directed to different servers in the cluster. Therefore, session data needs to be maintained externally to the Web servers. Duwamish Online stores session state in a separate database and all servers in the cluster have access to this database. To avoid having the database introduce a single point of failure, it's important to make the database server as reliable as possible by using failover clustering and a RAID array for disk storage.

### Failover Testing for a Cluster

In order to demonstrate how NLB works during failover, we ran a load test against a two-node NLB cluster with the Microsoft Web Application Stress Tool (http://webtool.rte.microsoft.com/). For more information about load testing a site, look for our upcoming article on performance testing Duwamish Online.

In this test, two Web servers and one database server were set up. Initially, both Web servers were running in an NLB cluster with equal load. After about a minute into the test, the network connection of Web server #2 was deliberately removed to simulate network or hardware failure of the server. About a minute later, the network connection was reestablished so that Web server #2 would converge automatically into the NLB cluster again.

Our test result shows that Web server #1 picked up all new requests within about 20 seconds after Web server #2 failed to operate (in this case, Web server #2 disconnected from the network). The overall system throughput of about 55 pages per second was maintained by Web server #1 after the failover occurred. This demonstrates that the remaining server within an NLB cluster continues to take on new requests automatically within a very short period of time.

It took about 15-20 seconds for Web server #2 to recover back to the original system throughput after the server was attached to the network again. Figure 4 shows that Web server #2 converged back to the NLB cluster within a very short period of time and continued to take on new requests evenly with Web server #1.

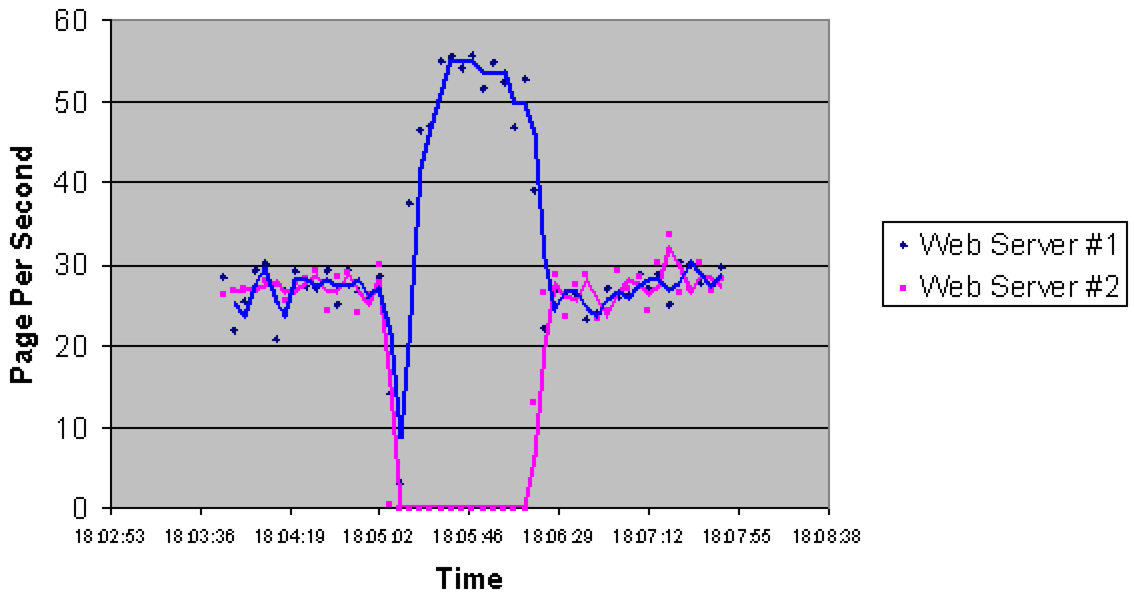

**Figure 4. System throughput of NLB cluster during failover**

### Scalability Testing of the Cluster

Figure 5 shows the results of load testing the Duwamish Online site with various numbers of Web servers in an NLB cluster. The servers used were inexpensive dual-processor, workstation-type machines. Note that the capacity scales linearly as we add machines to the cluster. As previously mentioned, the best resource utilization typically occurs with clusters limited to 16-24 servers—well below the upper limit of 32 supported by NLB. Larger clusters can be built by load balancing across multiple NLB clusters using either Round Robin DNS or load-balancing switches. NLB clusters of multiprocessor servers are also very scalable.

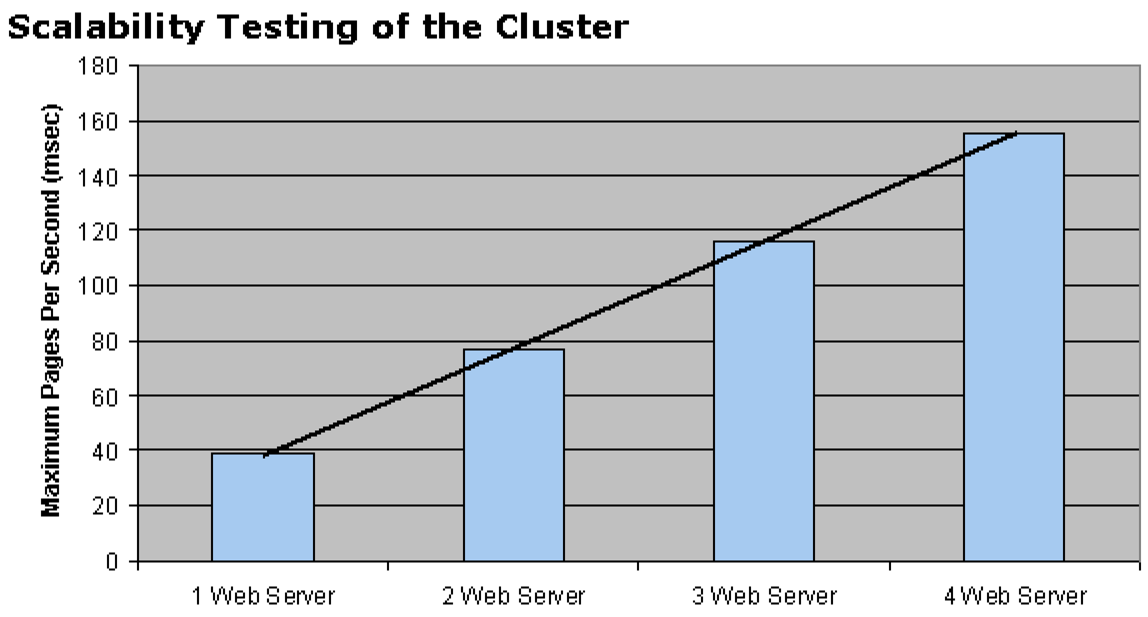

**Figure 5. Increase maximum system throughput by adding Web servers**

## Conclusion

It's very important, from a reliability and scalability standpoint, to be able to have multiple Web servers handling your Web site. In order to make use of multiple servers, you must have some sort of load balancing.

The simplest form is Round Robin DNS—but RRDNS has numerous technical limitations, the worst of which is that it has no way of detecting a slow or dead server, and will therefore send requests to servers regardless of whether it can handle them. However, RRDNS is sometimes successfully used for small, non-critical Web sites or, more commonly, for load balancing across a set of clusters.

Many sites—particularly those that aren't using Microsoft Windows NT Server or Windows 2000 on their Web servers—use a load-balancing switch to distribute requests. Load-balancing switches do a great job of distributing the load, but they're expensive and represent a single point of failure, so they can reduce reliability and availability of your Web application.

If you're running Windows 2000, you'll want to consider Windows 2000 Advanced Server Network Load Balancing. It doesn't add a single point of failure, doesn't require expensive hardware, and is often less expensive than a set of load-balancing switches. (There's a version called Windows Load Balancing Service (WLBS) for those still running Windows NT.)

## Appendix: How Round Robin DNS Works

When an application or a user requests a resource on the Internet, the Internet name of the resource can be used. For example, an application will use the name "msdn.microsoft.com" instead of an IP address. However, to locate the resource on the Internet, you need to know the IP address of the resource. Domain Name System (DNS) is the standard method of translating the Internet name of a resource into its IP address. The method of translating names into their corresponding IP addresses is called *name resolution*. For more information on DNS, see our article Setting Up a Domain Name System.

It's possible for a domain's DNS servers to give a different IP address (or, a different ordering of the set of possible IP addresses) each time it's queried. Each of the IP addresses points to a logically identical server that's equally capable of handling the request. And because different clients are routed to different machines (with different IP addresses), this gives us a primitive form of load balancing.

The main advantage of RRDNS is that it requires no additional hardware—you just set up your DNS server properly, and it works. However, there are several disadvantages that prevent many sites from using RRDNS for load balancing:

- The caching feature of DNS prevents complete load balancing because not every request that comes in will get its address directly from our DNS server.

  You can solve this issue by disabling caching, but doing so means that every resolution will have to be resolved by your servers, which is expensive and potentially slower for your users.

- The DNS server has no way of knowing if one or more of the servers in your Web farm is overloaded or out of service. So, the round-robin scheme will send traffic to all servers in turn, even if some are overburdened or dead. This is not exactly a wonderful user experience, although a user can click **Refresh** to try again (and have a random chance of succeeding, assuming caching doesn't occur).

Primarily due to this last issue, Round Robin DNS isn't used often (at least not independently) for large or mission-critical Web farms. But you can use RRDNS for load balancing in a Web farm with two or three servers—or you can use it to balance load across two or three server clusters, each of which is load balanced with one of the other load-balancing solutions mentioned in this article. (The chances that an entire cluster will fail are remote.)