

realtimepublishers.com<sup>tm</sup>

*The Definitive Guide<sup>tm</sup> To*

# Scaling Out SQL Server 2005

*Don Jones*

---

Chapter 7: Scale-Out and Manageability .....	144
Manageability Problems in a Scale-Out Environment.....	144
Monitoring .....	145
Maintenance .....	145
Management.....	148
Monitoring Solutions for Scale-Out.....	148
Microsoft Operations Manager.....	151
Third-Party Solutions.....	153
Symantec Veritas i <sup>3</sup> for SQL Server .....	153
Unisys Application Sentinel for SQL Server.....	154
ManageEngine Applications Manager.....	154
Nimsoft NimBUS for Database Monitoring .....	156
NetIQ AppManager for SQL Server.....	156
Maintenance Solutions for Scale-Out .....	158
Microsoft Windows Server Update Services .....	159
Microsoft Systems Management Server .....	160
Third-Party Solutions.....	161
ConfigureSoft Enterprise Configuration Manager.....	161
Diskeeper .....	162
ScriptLogic Service Explorer.....	163
Backup and Restore .....	164
Management Solutions for Scale-Out.....	165
Hardware Scenarios for Easier Scale-Out Management.....	167
Blade Computing .....	168
Storage Solutions .....	168
Summary .....	170

## **Copyright Statement**

© 2005 Realtimedpublishers.com, Inc. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtimedpublishers.com, Inc. (the "Materials") and this site and any such Materials are protected by international copyright and trademark laws.

THE MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. The Materials are subject to change without notice and do not represent a commitment on the part of Realtimedpublishers.com, Inc or its web site sponsors. In no event shall Realtimedpublishers.com, Inc. or its web site sponsors be held liable for technical or editorial errors or omissions contained in the Materials, including without limitation, for any direct, indirect, incidental, special, exemplary or consequential damages whatsoever resulting from the use of any information contained in the Materials.

The Materials (including but not limited to the text, images, audio, and/or video) may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, in whole or in part, except that one copy may be downloaded for your personal, non-commercial use on a single computer. In connection with such use, you may not modify or obscure any copyright or other proprietary notice.

The Materials may contain trademarks, services marks and logos that are the property of third parties. You are not permitted to use these trademarks, services marks or logos without prior written consent of such third parties.

Realtimedpublishers.com and the Realtimedpublishers logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners.

If you have any questions about these terms, or if you would like information about licensing materials from Realtimedpublishers.com, please contact us via e-mail at [info@realtimedpublishers.com](mailto:info@realtimedpublishers.com).

[**Editor's Note:** This eBook was downloaded from Content Central. To download other eBooks on this topic, please visit <http://www.realtimepublishers.com/contentcentral/>.]

## Chapter 7: Scale-Out and Manageability

In addition to bringing a solution to larger workloads, scale-out solutions also bring a unique set of problems to your environment, primarily in the realm of manageability. These are similar to the problems encountered by administrators building a Web farm: With so many additional servers handling the same application, how do you keep content synchronized across the farm? When changes are made to server configuration standards, how can you accurately deploy those changes across the farm? How can you efficiently monitor the health of the servers in the farm? In this chapter, I'll focus on solutions that can help solve the manageability problems in a SQL Server scale-out solution.

### Manageability Problems in a Scale-Out Environment

The manageability problems in a scale-out environment are more than just having to manage multiple servers, it's the fact that those multiple servers all form a single application. In other words, you have to somehow manage the servers almost as a single unit, even though they're distinct, independent units, so that you can maintain the integrity and functionality of the overall application. That's a problem made more difficult in a SQL Server scale-out than even in a Web farm; with Web farms, taking a single server offline isn't a big deal, because the other servers in the farm do the same thing and can pick up the slack. In most SQL Server scale-out solutions, however (such as a federated database), each individual server is a crucial element of the overall application. In the next three sections, I'll explore some of the specific issues with manageability in a scale-out environment, so that you can clearly understand the challenges that you face.



When it comes to solutions, this chapter will focus almost entirely on SQL Server 2005, rather than SQL Server 2000. While most of the add-on tools from Microsoft and third parties are available for SQL Server 2000, the built-in manageability capabilities I'll describe are almost entirely unique to SQL Server 2005.

## Monitoring

There are a few major goals of server monitoring, and it's important to really spell them out in order to understand how they're impacted by a scale-out scenario:

- **Health.** One main goal of monitoring is to keep an eye on server—or, more accurately, *application*—health. *Health* is differentiated from *performance* by the level of context it uses. For example, monitoring CPU performance requires very little analysis; the CPU is what it is, and if performance is sitting at 60% utilization, then that's your performance metric. There's no context; the utilization is simply 60%. *Health*, however, places that number into context, and answers the question, “is this server (or application) healthy or not?” In other words, is 60% processor utilization—along with other performance metrics—good or bad?
- **Availability.** One particularly important goal of monitoring is to measure the availability of a server (or application), and to notify the appropriate people if the server (or application) becomes unavailable.
- **Trending.** Another important goal of monitoring is to develop trend reports, which help predict future workload requirements based on past workload and observed growth.

A scale-out solution makes these goals more difficult to achieve. For example, if you have a federated database consisting of three SQL Server computers, the health of the *application* is governed by the combined health of all three servers. You can't simply take an average of performance metrics; one server consistently running at 100% utilization, for example, will drag down the application performance even if the other two servers are only at 40% utilization. You essentially need a solution that can monitor metrics of the application itself—total response time to key queries, for example—rather than individual servers. However, you still do need to monitor the health of individual servers, because some issues—such as poor disk throughput or high memory utilization—can be an indicator of server-specific issues that you can troubleshoot and address appropriately.

## Maintenance

Maintenance is one of the most complex and difficult areas of a scale-out solution. Maintenance consists of ongoing tasks designed to keep servers (and the application) healthy, secure, and available, such as:

- Applying hotfixes and patches
- Applying service packs
- Scanning for viruses and other malware
- Inventorying hardware and software
- Defragmenting hard disks or databases
- Maintaining security settings
- Rebuilding indexes
- Updating database statistics used by the SQL Server query optimizer



I'm not including hardware-level maintenance, which typically involves shutting a server down, in this list because that type of maintenance is always conducted per-server. In other words, if you need to upgrade the memory in four SQL Server computers, it's going to require physical service on all four servers. Software-level maintenance, however (such as the items in the above list), can often be conducted at an application level by using tools that help to automatically apply the maintenance task across all of the application's servers.

I categorize these tasks into two broad areas: Operating system-level, and SQL Server-level. Operating system-level maintenance involves taking care of Windows itself, and the operating system and SQL Server tasks sometimes parallel one another. For example, applying patches is something you'll do for both Windows and SQL Server; rebuilding indexes is a SQL Server-specific task, while inventorying hardware typically applies only to Windows.

Some of these maintenance tasks—such as patch management or defragmentation—are difficult enough on a single server. However, the need for consistency across all the servers in an application makes these tasks doubly difficult in a scale-out scenario. For example, if you need to make changes to security settings, it's absolutely essential that the same change be made, at nearly the same time, to all of the servers in the solution. Otherwise, users could experience inconsistent results.

Many of these maintenance tasks are time-consuming, as well. For example, keeping track of index status—a monitoring task—and rebuilding indexes when necessary—a maintenance task—requires a lot of continual time and attention from valuable administrative resources. In fact, one of the major objections to any solution which entails adding more servers to the environment—such as a scale-out solution—is the amount of additional administrative overhead the new server will require simply due to its existence. In order for scale-out solutions to be feasible, they must not only function, but they must also create as little additional administrative overhead as possible.

### The DSI Solution

The biggest problem in any scale-out solution is the concept of managing a group of servers as a unit, rather than managing individual servers. For decades, IT management has been performed more or less at the server level; solutions that manage groups of servers as a unit are rare. Microsoft Application Center 2000 was one such solution, allowing you to make a change to one Web server and automatically replicating that change to every server in a Web farm. However, Application Center 2000 was specific to Web servers.

Microsoft's long-term solution to the problem is their Dynamic Systems Initiative, or DSI. A core part of DSI is the System Definition Format, or SDF, an XML format that describes a configuration. In its fully-realized implementation (which is still years away), DSI will help better manage application—rather than server—configurations from initial provisioning throughout the application lifecycle.

It's supposed to work something like this: When you decide you need a new SQL Server application (for example), you'll use a configuration tool to create your desired application configuration. This may include installing and configuring SQL Server, IIS, and a number of other components. You won't actually perform these tasks; you'll just specify them. The result is an SDF file describing exactly what a server in your application should look like. You'd feed that SDF file to a provisioning tool (perhaps a successor to the current Windows Automated Deployment System), which would actually install and configure the necessary software on a new server for you.

DSI would then ensure that your intended configuration remained in place. For example, if another administrator modified the server's local firewall settings, DSI might reconfigure the server—automatically—back to the settings required by the SDF file. If you need to make an approved change to the server's configuration, you'd make the change *in the SDF file*, using some sort of configuration tool. The change to the file would trigger DSI—which would be implemented throughout Windows, IIS, SQL Server, and any other products you're using—to physically reconfigure the server to match the revised file. In essence, the SDF file serves as your configuration standard, and DSI works to automatically configure servers to match that standard at all times.

You can see how some elements of Microsoft's current product line—notably Systems Management Server (SMS) and Operations Manager (MOM)—might evolve over time to encompass some of DSI's functionality. And you can also see how DSI would make managing multiple servers easier: You simply use DSI to initially provision however many servers you need according to a single standard. Any changes that need to be made are made *once*, to the standard, and DSI implements those changes on any servers which are set to follow that standard. It's true policy-based management rather than server-based management.

As I mentioned, the full vision of DSI won't be realized for years to come, but understanding that DSI is the eventual goal can help you make smarter decisions about management techniques, technologies, and practices now.

Other types of maintenance tasks have very unique problems in a scale-out solution. For example, backing up servers and databases is a common maintenance task that's made more complicated in a scale-out solution. You're faced with two problems: First, the task of backing up increasingly-large databases, which is difficult enough in and of itself; and second, the task of ensuring your entire application—no matter how many servers or databases are involved—is backed up as a unit (to the best degree possible), so that any major recovery effort can bring the entire application back online in a consistent, usable state.

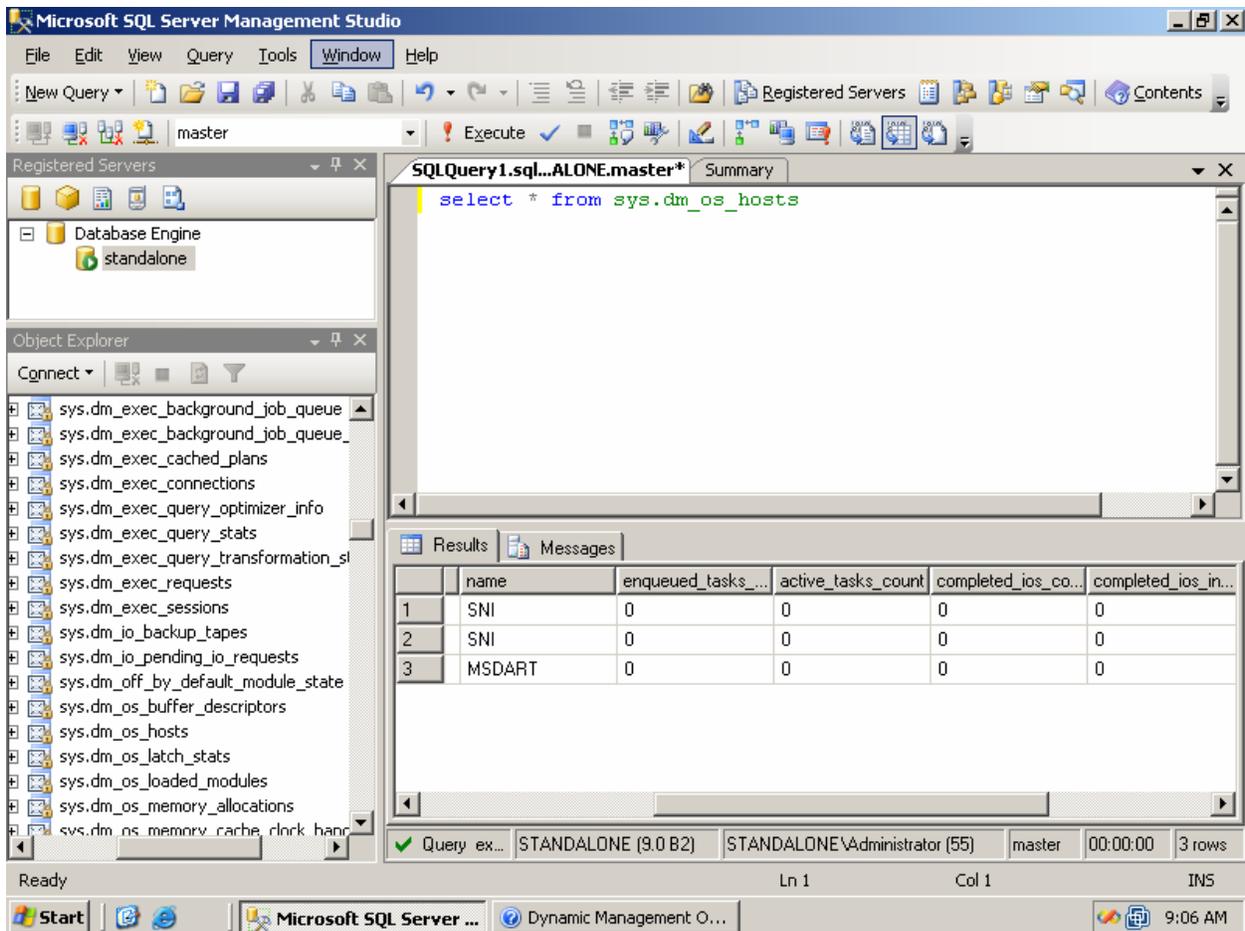
### **Management**

Management is the process of making periodic changes to your servers or applications. It's closely related to maintenance; maintenance, however, generally consists of the management tasks that you can always expect to occur in some quantity over any given period of time. Patch management, for example, is a true maintenance task: You know it's going to be necessary. I distinguish true management tasks as those which aren't always predictable, but which occur in response to some business condition. Reconfiguring servers to meet a new business need, for example, is a one-time task that isn't performed on a regular basis.

Management tasks face many of the same challenges as maintenance tasks: Changes need to be applied consistently, and more or less simultaneously, across all of the servers in the solution. If anything, management tasks tend to involve more sweeping, major changes, meaning that mistakes in performing these tasks can have more serious consequences. Unfortunately, today's technologies tend to still focus on server-based management, making application-based management difficult.

### **Monitoring Solutions for Scale-Out**

SQL Server 2005 introduces a new feature called Dynamic Management Views (DMVs). Physically implemented as actual database views (contained in the Master database), DMVs are designed to sum up performance and other management information across the entire server. Much of the information in DMVs would otherwise only be available on a per-process basis, making it difficult to get an accurate server-side view of this information. Figure 7.1 shows a DMV, which can be executed—just like any other view—in Management Studio.



**Figure 7.1: Examining a DMV.**

More than 70 DMVs are included with SQL Server, including:

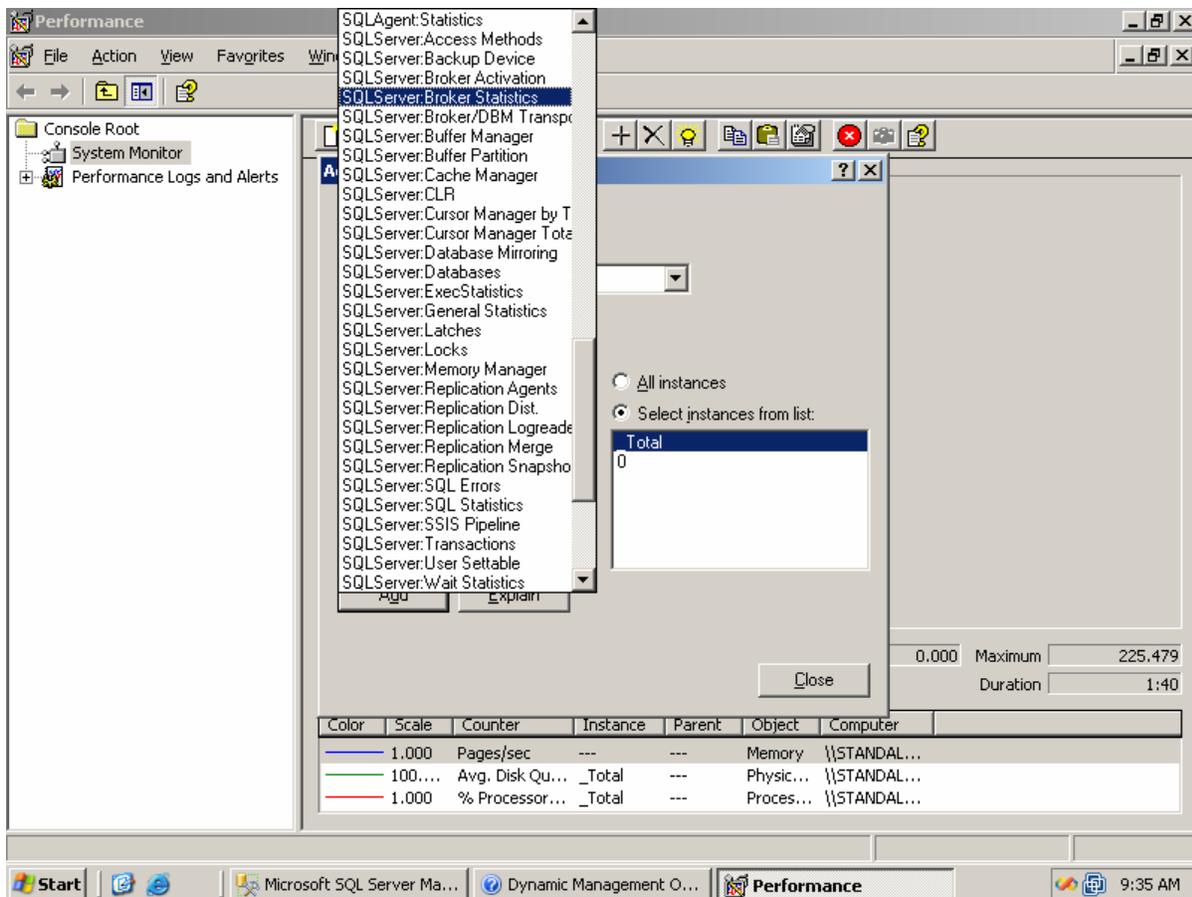
- Broker connections
- Child instances
- CLR application domains
- CLR loaded assemblies
- Index usage statistics
- Partition statistics
- Cached execution plans
- Query statistics
- Backup tapes
- Pending I/O requests
- Hosts
- Loaded modules

- Memory allocations
- Cache entries
- Threads
- Wait statistics
- Replication articles
- Transaction locks
- Active transactions

And so forth. In terms of single-server management, DMVs make a wealth of performance and health data available to an administrator. However, they don't do anything to provide consolidated data across a group of servers participating in a scale-out solution. There is an application for DMVs in scale-out solutions, however: If you consider how difficult it would be to obtain the information from a DMV on a single server, without using the DMV, then you can imagine how hard compiling that information would be for multiple servers. Although figuring out index statistics for multiple servers would require you to execute a DMV on each server, that's a better solution than trying to assemble that information *without* the DMV.

 You could write a stored procedure that queried information from multiple servers' DMVs to present the information in a somewhat consolidated query result.

From a more traditional performance perspective, SQL Server provides performance objects and counters for Windows' built-in Performance Monitor console. As Figure 7.2 shows, dozens of objects and hundreds of counters are available, allowing you to measure even the most detailed portions of a SQL Server computer's performance.

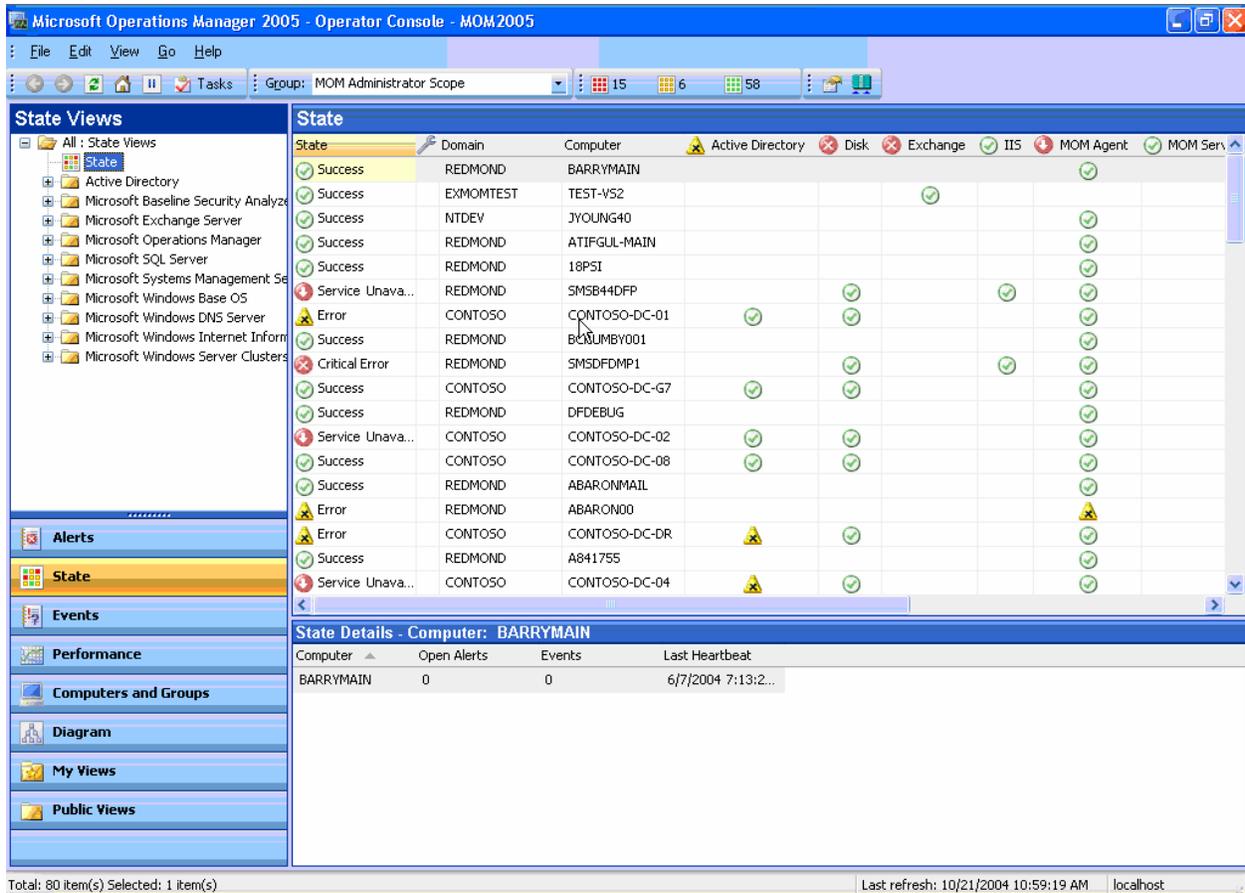


**Figure 7.2: Monitoring SQL Server in the Performance console.**

Again, however, this is just per-server performance monitoring. For somewhat higher-level monitoring, you'll need to turn to other products.

### **Microsoft Operations Manager**

Microsoft Operations Manager (MOM) is essentially a health monitoring solution. At its core, MOM collects performance data from individual servers, and then places that data into a context to create a “healthy” or “not healthy” indication. This context is provided by *management packs*, which codify Microsoft’s expertise in running their products, and provide the “intelligence” behind the indication. The management packs tell MOM what performance counters to watch, what performance levels are considered normal, and what performance levels are considered indicative of a possible failure or a pending problem. Microsoft provides management packs for most of its server products, including SQL Server. MOM can help make multi-server management (such as in a scale-out solution) easier by providing consolidated views that list each server and its health (see Figure 7.3). Specific columns provide access to application health on a per-server basis, such as allowing you to see what SQL Server instances, Exchange servers, IIS servers, and so forth are, or are not, considered healthy.



**Figure 7.3: MOM provides a consolidated server health readout.**

MOM doesn't have any built-in capability for monitoring specific applications. However, you *can* build your own “management packs,” of a sort, using MOM's capabilities. This would allow you to, for example, configure MOM to monitor an entire application consisting of multiple servers, and to test response times to (for example) specific queries or other operations. You would define the thresholds of what was considered healthy or not, allowing MOM to monitor your application—rather than just individual servers—and provide feedback about the state of the application's health.

For more information about MOM, visit [www.microsoft.com/mom](http://www.microsoft.com/mom). Note that, as of this writing, management packs specific to SQL Server 2005 have not yet been made available to the public.

### Third-Party Solutions

The third-party software market is a rich source of solutions for monitoring SQL Server applications, particularly in a scale-out environment.

 These solutions are simply examples; many manufacturers offer similar solutions in the same categories.

### Symantec Veritas i<sup>3</sup> for SQL Server

Veritas i<sup>3</sup> is an application service management solution, which is designed to measure the end performance of entire applications, rather than focusing on the performance of the servers which comprise those applications. The tool allows you to monitor, analyze, and tune various elements of a SQL Server environment by capturing performance metrics from each tier of the application, including the client tier (which is ultimately all that matters, since the client tier is what your end users are interacting with directly). The tool can measure the response time of various operations at each level, from the client all the way down to specific SQL Server statements, allowing you to more quickly pinpoint the cause of a problem—a capability you'll need in large scale-out solutions. Figure 7.4 shows how performance for an operation is broken down by tier (each band of blue represents the time an individual tier or element contributed to the overall response time).



Figure 7.4: Analyzing application, rather than server, performance.

 For more information, visit <http://www.veritas.com/Products/www?c=product&refId=317>

The product also provides resource consumption information on a per-table basis, which can help you identify tables that need to be distributed, or kept together, in a scale-out scenario.

## Unisys Application Sentinel for SQL Server

Unisys' product is a basic SQL Server performance management tool, providing performance monitoring and prescriptive guidance for improving performance. The tool is designed for SQL Server installations running on Unisys ES7000 servers. If you're using these servers in a scale-out solution, then Application Sentinel can provide basic performance monitoring capabilities.

 For more information, visit [www.unisys.com](http://www.unisys.com).

## ManageEngine Applications Manager

Another performance monitoring tool, ManageEngine Applications Manager provides similar capabilities to MOM, in that it allows you to define “healthy” and “problem” thresholds and be alerted when server performance exceeds the thresholds. As shown in Figure 7.5, you can configure monitor groups, which allow you to more easily monitor an entire scale-out solution from a single view, including supporting services such as Web servers. Other capabilities include reports and graphing for trend analysis (an important component of the long-term monitoring for any scale-out solution), and the ability to track server activity by user, helping to identify problem SQL statements and relate them to a particular thread of activity.

The screenshot displays the ManageEngine Applications Manager 6 interface. At the top, there is a navigation bar with tabs for Home, Monitors, Alerts, Reports, Support, and Admin. Below this is a search bar and a list of navigation links. The main content area is titled "Demo Monitor Group" and contains several sections:

- Monitor Group Information:** A table showing details for the "Demo Monitor Group", including its name, health status (critical), root cause (three application health issues), type (Monitor Group), description (For WebEx Demo), owner (root), creation and modification dates, and the number of monitors (14).
- Today's Availability:** A pie chart showing 84.18% uptime (green) and 15.82% downtime (red). A legend indicates "Downtime 15 Hrs 17 Mins" and "Uptime 2 Hrs 52 Mins".
- Monitors Present in this Monitor Group:** A grid of eight individual monitor cards. Each card shows the monitor name, type, today's availability pie chart, response time, and current health status.
  - AppManager! Help:** HTTP-URL Sequence, 100.00% availability, response time -.
  - AppManager!:** HTTP-URLs, 100.00% availability, response time 1741 ms.
  - app-linux4.india.adventnet.com Tomcat-serv...:** Tomcat Server, 100.00% availability, response time 5 ms.
  - app-linux4.india.adventnet.com MYSQL-DB-se...:** MySQL Database Server, 100.00% availability, response time 3 ms.
  - app-w2k1.india.adventnet.com MSSQL-DB-server:** MS SQL Database Server, 96.75% availability, response time -.
  - app-w2k1.india.adventnet.com ORACLE-DB-server:** Oracle Database Server, 100.00% availability, response time 1130 ms.
  - app-linux4.india.adventnet.com WEB-server:** Web Server, 100.00% availability, response time 11 ms.
  - smtp\_MAIL-server\_25:** Mail Server, 100.00% availability, response time 150 ms.

Figure 7.5: ManageEngine Application Manager.

For more information, visit [http://manageengine.adventnet.com/products/applications\\_manager/sql-server-management.html](http://manageengine.adventnet.com/products/applications_manager/sql-server-management.html).

## Nimsoft NimBUS for Database Monitoring

In addition to monitoring basic server performance metrics, NimBUS for Database Monitoring also measures query response times and transaction rates, which can provide a more accurate view of overall *application* (as opposed to per-server) performance in a scale-out solution. This type of monitoring allows you to establish, and monitor, service level agreements (SLAs) for response times to end users of your scale-out application. NimBUS can also save performance reports for archival purposes, providing a good source for trending data, allowing you to see how workload demands on your application change over time, and allowing you to make better decisions about future scale-up or scale-out engineering. Figure 7.6 shows NimBUS' setup dialog, which allows you to determine which performance metrics you'll view. As shown, metrics such as active users, transactions per second, and so forth are excellent for trending purposes.

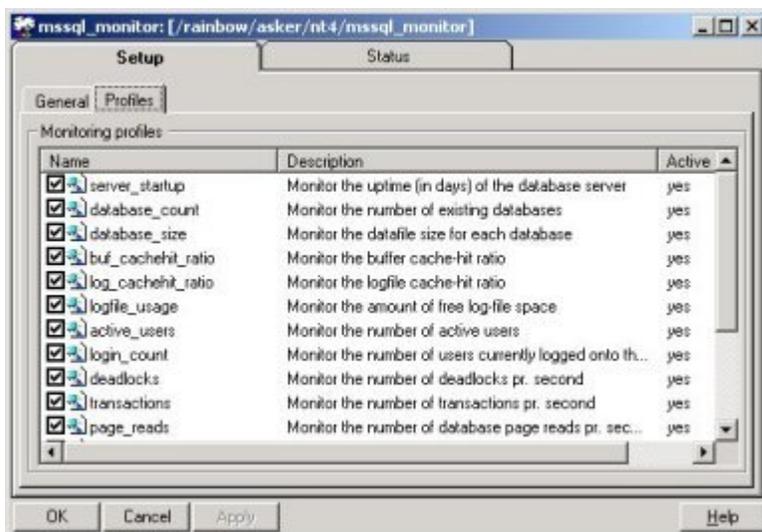


Figure 7.6: Configuring NimBUS for SQL Server monitoring.

For more information, visit <http://www.nimsoft.com/environments/databases.shtml>

## NetIQ AppManager for SQL Server

AppManager occupies an interesting place in the market. It's actually the technology on which MOM is built, making the two products similar in base capabilities. However, AppManager for SQL Server is much more SQL Server-specific, tracking performance down to the SQL statement level and providing detailed statistical analyses of performance metrics. AppManager provides trending on a per-server basis to help you figure out what hardware upgrades might be required or advisable in the future.

Although monitoring performance on an application level is preferable to monitoring performance on a per-server level (since it's ultimately the application's performance you care about), the same is not true for trending. In a SQL Server scale-out environment, server load is rarely distributed evenly. Therefore, metrics like CPU utilization, memory utilization, I/O throughput, and so forth *are* useful for trending on a per-server basis, since these trends can help alert you to areas which are becoming bottlenecks on a particular server, and which therefore may need to be upgraded in order to maintain application performance.

Like MOM, AppManager isn't specifically configured to monitor overall application performance, but you can set up your own performance measurements within AppManager to measure key response times.

Applications like MOM and AppManager can't typically measure direct client application response times. However, you can get a good measurement of overall application performance by making critical queries accessible through Web pages or Web services. MOM, AppManager, and most other performance applications *can* measure Web request response time, and that response time would include (and in fact would primarily consist of) the query response time. While raw query response time isn't quite the same thing as overall application performance, measuring the response times for queries that really impact your application (such as queries based on a distributed partitioned view or other scale-out element) provide a good indicator of application performance.

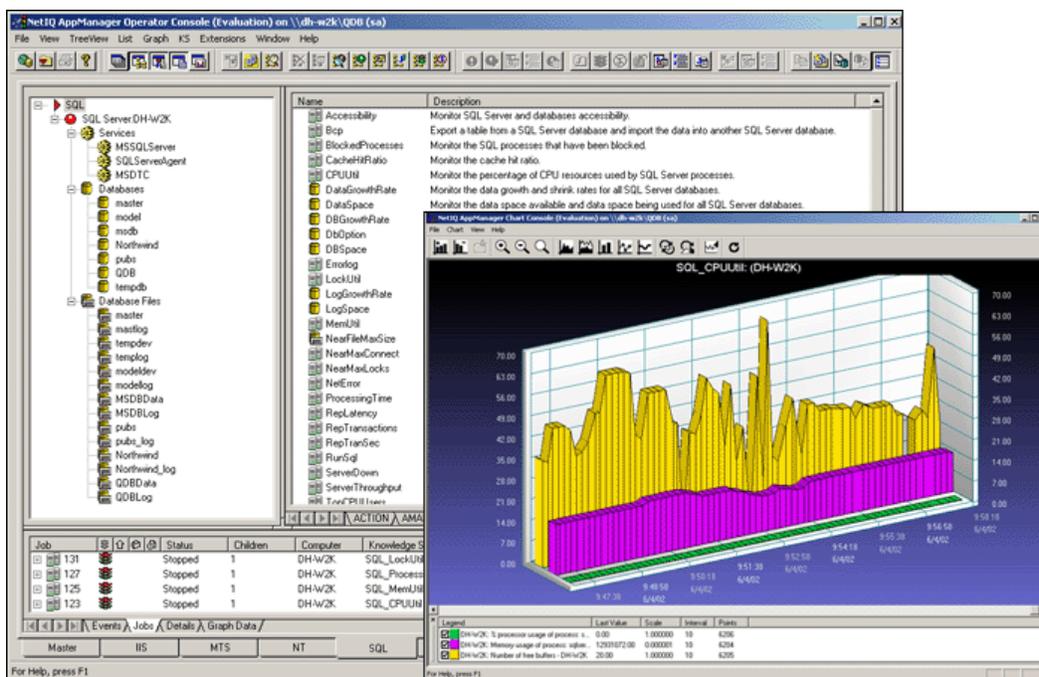


Figure 7.7: Using AppManager to monitor SQL Server performance.

For more information, visit <http://www.netiq.com/products/am/modules/sql.asp>.

## Maintenance Solutions for Scale-Out

SQL Server 2005 is no slouch when it comes to improved maintenance options. In fact, many of its new maintenance features are extremely helpful in a scale-out solution, even if they're not specifically designed for scale-out. Perhaps one of the most useful maintenance features in SQL Server 2005 is online indexing.

In prior versions, most indexes operations required that the database or table be made unavailable to users. For example, building a new index, or rebuilding an existing index, couldn't be performed while changes were being made to the database. Online indexing changes this, allowing indexes to be built or rebuilt while continuing to allow changes to the underlying data. Essentially, SQL Server does the initial index build (or rebuild) on a snapshot of the data, while keeping track of data page changes that occur as the build progresses. When the build is finished, SQL Server incorporates the data page changes into the index, providing an up-to-date index. This feature isn't specifically designed for scale-out scenarios; it's an availability option that helps keep any database up and running. However, it does come particularly in handy in scale-out solutions. Keep in mind that, in a scale-out solution, the entire database is spread across multiple servers (in some fashion, depending on how you've scaled out). Making one table on one server unavailable for indexing could make the entire *application* unavailable; online indexing allows index maintenance to occur while keeping the application online and accessible.

Another important maintenance feature is online restore. In past versions, SQL Server required an entire database to be taken offline in order to restore it from backup. Again, in a scale-out solution this can present difficulties because an entire application, spread across multiple servers, would be made unavailable. With online restore, the database continues to be accessible, with only the data actually being restored made unavailable. This helps to improve up-time, even when maintenance tasks like data restores are underway,

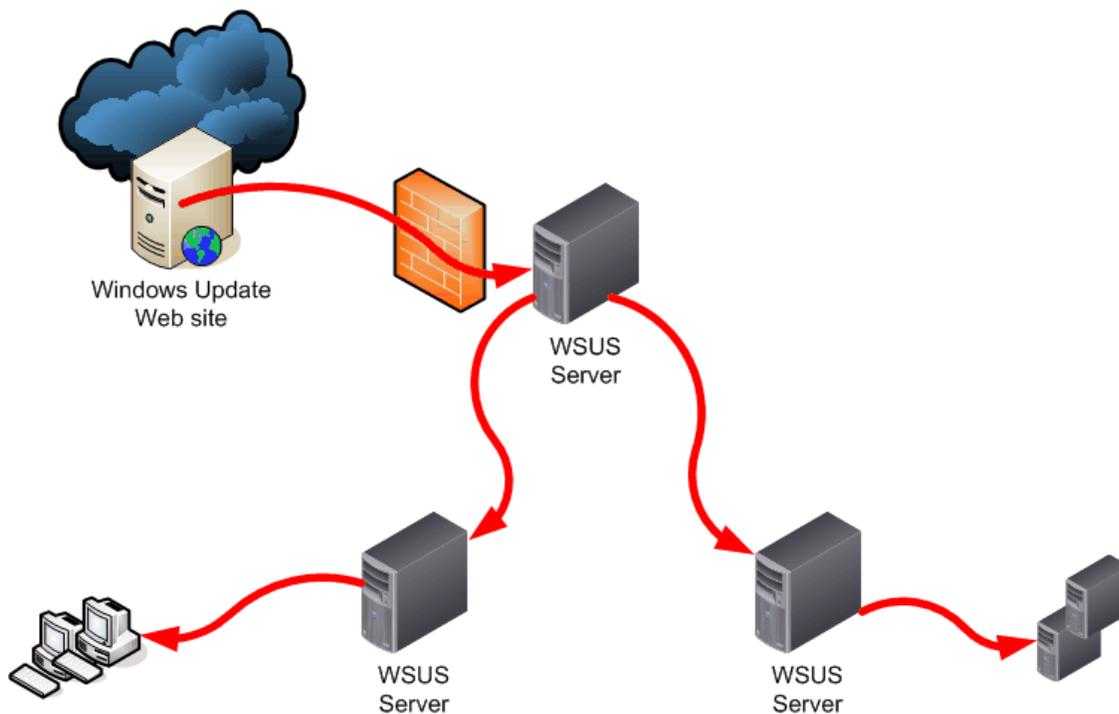
Table partitioning was added to SQL Server 2005 to help improve maintenance and management. Essentially, table partitioning allows a table to be partitioned across multiple files, yet still managed as a single unit. This allows an administrator to, for example, spread a table's contents across multiple *disks* (since each file can be located on a different disk), thus spreading the table's workload across storage devices. Because SQL Server often bottlenecks at disk access, spreading the table across disks can help keep data flowing into memory more freely, improving performance. But this is really a scale-up capability, not scale-out, allowing SQL Server to do more on a single server than it otherwise could. Table partitioning doesn't include the ability to partition a table automatically across *servers*, which would be a true scale-out capability. You can of course *manually* partition tables across servers in a federated database, but you're left managing each partition as an independent table, rather than being able to manage them as a unit.

SQL Server still leaves you with maintenance difficulties in a scale-out solution: Patch management, overall systems management, disk defragmentation, security maintenance, and even backups can be difficult when your data is spread out across multiple servers. Fortunately, there are a number of products from Microsoft and other companies that can help make maintenance in a scale-out environment easier.

## Microsoft Windows Server Update Services

Formerly known as Software Update Services (SUS) and, briefly, Windows Update Services (WUS), Windows Server Update Services (WSUS) is a corporate version of the Microsoft Update Web site. Essentially, WSUS allows you to take control of Microsoft patch management for not only Windows itself, but also for other Microsoft business products such as Office and, of course, SQL Server.

WSUS is designed to be installed in a hierarchical fashion, as shown in Figure 7.8. Updates are downloaded from the Windows Update Web site by a top-level WSUS server, where updates can be reviewed and approved. Once approved, updates are downloaded by downstream WSUS servers (typically, at least one per major geographic location on your network), and from there deployed to client computers and servers. Updates can be approved for groups of computers, and groups can be constructed both at the WSUS server and centrally through Active Directory (by assigning group membership through Group Policy).



**Figure 7.8:** Deploying WSUS in a hierarchical fashion.

The key with WSUS is the client-side Automatic Updates client software, which can be configured (again, via Group Policy) to look for updates on the local WSUS server rather than the Microsoft Update Web site. Automatic Updates can be configured to look for updates automatically, on a regular basis, and to automatically download and install updates. This capability helps to remove patch management as an active administrative task and instead makes it passive; adding multiple servers in a scale-out solution no longer requires the additional overhead of managing patches on additional servers.



A large number of third-party solutions also exist to help make patch management easier. In most cases, however, WSUS is all you need, and it's completely free. Unlike prior versions (SUS), WSUS can provide updates for most of Microsoft's business products, including SQL Server.

### Microsoft Systems Management Server

While WSUS can help ensure that your systems have the latest patches and service packs, systems management goes a bit further than that. You'll also need to track hardware and software inventory, deploy other types of software (both new and updated), and so forth; Systems Management Server (SMS) can help make those maintenance tasks easier, even in a large scale-out environment. In fact, to avoid the need to manage multiple management and maintenance systems, SMS can even integrate with SUS (as Figure 7.9 shows) or WSUS, providing you with a single toolset for deploying anything, whether it's patches or entire software applications.

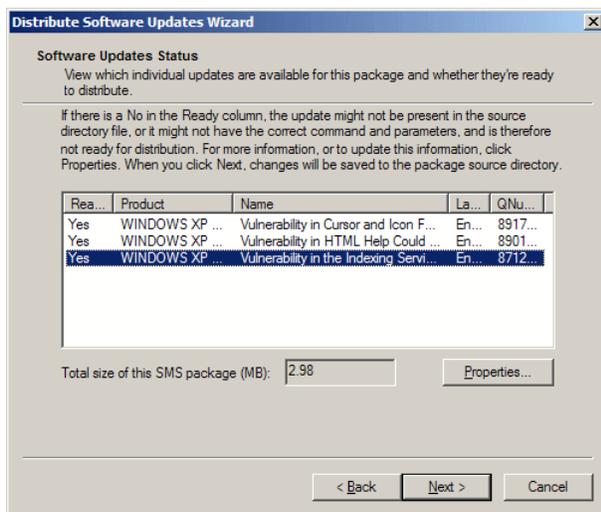


Figure 7.9: Integrating SUS with SMS.

However, the current version of SMS is designed primarily for inventorying and software deployment; it isn't designed to *push* configuration changes to managed servers, a capability that's sorely needed in scale-out scenarios to help maintain consistent configurations across multiple servers. For example, SMS can't help manage password changes for SQL Server service accounts, an absolutely crucial capability in managing multiple servers. Fortunately, a number of third-party solutions exist to help with various critical maintenance tasks.

### Third-Party Solutions

Third-party software developers can often provide point solutions that help solve specific problems, particularly in a scale-out environment where you're managing multiple servers and trying to achieve a high degree of configuration consistency.

 These solutions are simply examples; many manufacturers offer similar solutions in the same categories.

### ConfigureSoft Enterprise Configuration Manager

ConfigureSoft Enterprise Configuration Manager (ECM) is a complement to Microsoft SMS (although ECM operates just fine by itself). Rather than simply inventorying computers' hardware and software, ECM also inventories their configuration, and can analyze those configurations for compliance with a standard you define. Figure 7.10, for example, shows a compliance report that indicates how well computers on the network comply with your antivirus software policies. Although ECM doesn't collect much in the way of SQL Server configuration (that data being stored, for the most part, in the Master database within SQL Server itself, rather than in a more accessible location like the Registry), ECM can help to better manage the Windows-specific configuration settings within a scale-out solution.

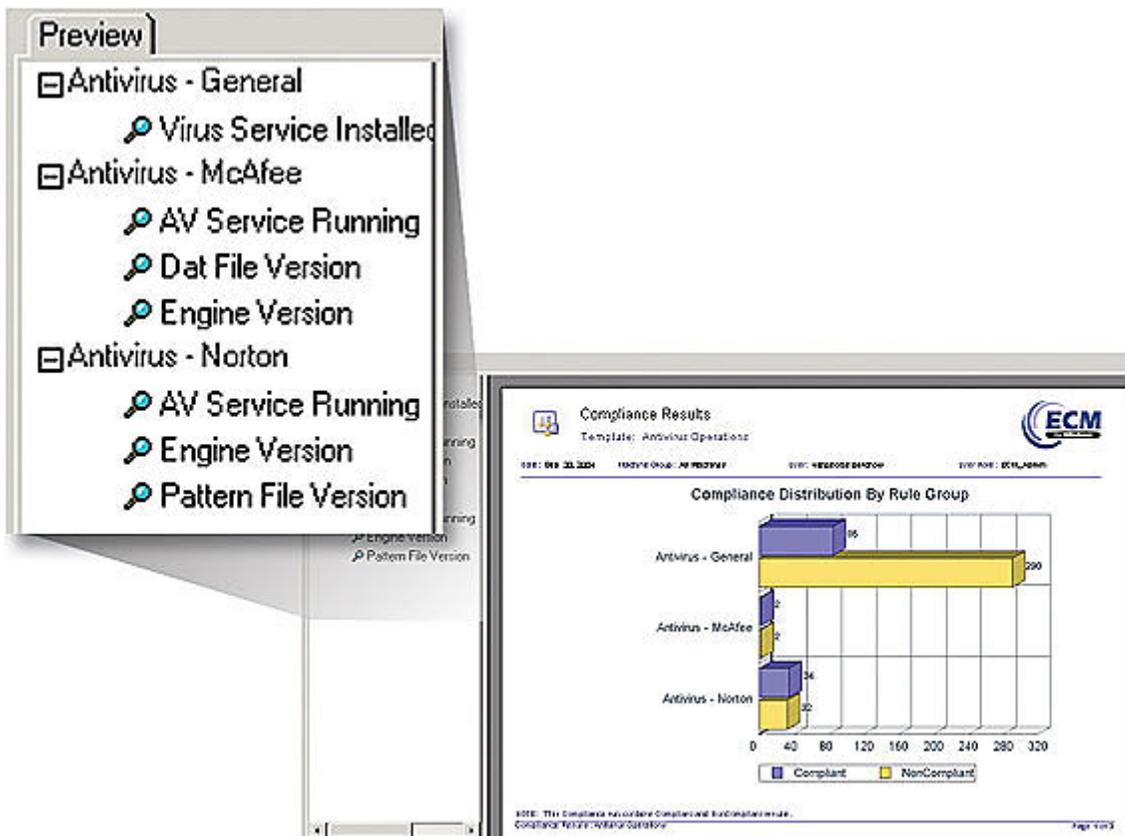


Figure 7.10: Viewing compliance reports in ECM.

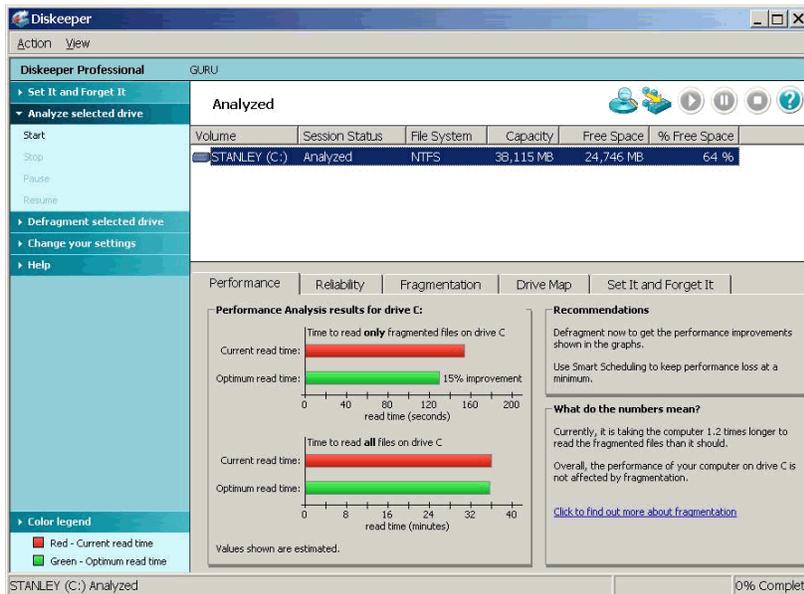
 For more information, visit [www.configuresoft.com](http://www.configuresoft.com).

## Diskeeper

Disk defragmentation affects SQL Server performance as much as any other application. SQL Server actually deals with two types of defragmentation: Physical, and in-database. SQL Server deals with in-database defragmentation on its own, reorganizing pages to keep data contiguous. Periodically compacting databases can help maintain their performance, especially in online transaction processing (OLTP) databases with frequent row additions and deletions. Physical defragmentation, however, refers to the database file itself becoming non-contiguous across the server's storage devices. Software like Diskeeper can help reorganize these files, and can be centrally managed to help reduce defragmentation on the servers in a scale-out solution. As Figure 7.11 shows, Diskeeper can analyze defragmentation and tell you how much slower disk access is as a result of defragmentation (in the example shown, disk access is almost 25% slower).

Diskeeper is smart enough not to try and defragment open files, which presents a special challenge for database files, since they're *always* open. You will need to close the databases in your scale-out solution in order to properly conduct a disk-level defragmentation. However, you can also take steps in advance to reduce or even eliminate disk-based defragmentation of your database files:

- Prior to creating your databases, thoroughly defragment the server's disks.
- Create the database with a large enough initial size to handle near-term growth. This ensures that the database file occupies contiguous disk space and that it contains enough empty room to support database expansion.
- Once created in a contiguous disk space, the database file cannot become defragmented (at this disk level, at least) until the database fills and needs to expand. At that point, you should again defragment the server's disks to provide sufficient contiguous free space and expand the database manually to a size that will accommodate all near-term growth.



**Figure 7.11: Managing defragmentation with Diskeeper.**

You can still periodically defragment server disks while SQL Server is running, provided your solution knows to leave SQL Server's open database files alone (in other words, treat them as *unmovable*, in much the same way that the Windows pagefile is usually treated). Diskeeper and similar solutions can be set to automatically defragment on a regular basis, helping to make this importance maintenance task passive, rather than requiring your active participation.

For more information, visit <http://www.diskeeper.com>.

## ScriptLogic Service Explorer

One of the most important and most often-overlooked maintenance tasks in any environment is password maintenance, particularly of service accounts, since these accounts aren't required by Windows to change their passwords on a regular basis as users are. In a scale-out solution, where servers must have valid credentials with which to communicate with one another, consistent service account configuration is absolutely essential.

ScriptLogic Service Explorer (see Figure 7.12) can be used to examine service account configuration and reconfigure service accounts automatically. By regularly using a tool like Service Explorer, you can ensure that passwords on service accounts remain fresh and uncompromised, and that all SQL Server services across your scale-out solution are consistently configured (a key to making SQL Server replication function properly in many scenarios).

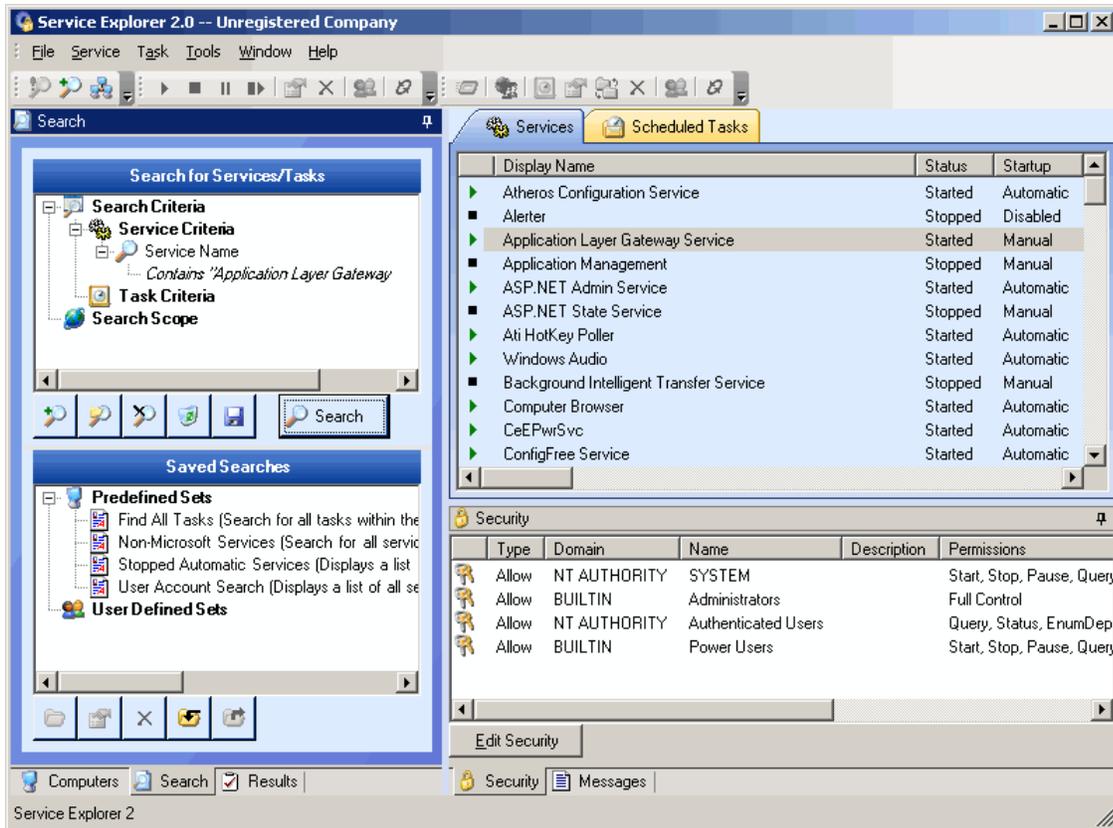


Figure 7.12: ScriptLogic Service Explorer.

For more information, visit <http://www.scriptlogic.com>.

## Backup and Restore

While SQL Server has perfectly adequate built-in backup and restore capabilities, scale-out solutions often require more flexible and powerful backup capabilities. For example, if you have a large enough database that scale-out is something you're examining, then you already know how difficult it is to grab a backup of the entire database in any kind of reasonable time. Once that data is scaled out across multiple servers, the task becomes even more difficult, especially if your goal is to periodically take a backup of the entire application data set—simply getting all of the servers to be in a consistent location with regard to data updates is often impossible. Enterprise backup solutions—like VERITAS Backup Exec or Computer Associates' BrightStor, simply aren't designed to coordinate backup activities across SQL Server computers in a scale-out solution. However, high-end storage systems often *can* make backup and restore easier, and it's something I'll discuss later in this chapter, under "storage solutions."

## Management Solutions for Scale-Out

Managing a scale-out solution requires special attention to consistency and caution; a single mistake on a single server can render an entire application virtually useless. Fortunately, SQL Server itself offers a number of built-in technologies that can help make management easier.

SQL Server Agent, SQL Server's built-in automation tool, is capable of targeting jobs to multiple servers. Essentially, that means you can create a single job, which can contain any number of steps, and then have that job pushed out to, and executed on, any number of servers. The job's status from each server rolls back up to the server on which the job was originally created. For tasks which can be performed through SQL Server Agent jobs (which can be pretty much anything within SQL Server), this is an excellent tool for minimizing the overhead of managing multiple servers to a consistent state.



Multi-target jobs were available in SQL Server 2000, as well as in SQL Server 2005.

For more complex operations, you can create your own management tools and scripts using SQL Management Objects (SMO), a completely managed application programming interface upon which SQL Management Studio itself is built. SMO is a programmatic way of controlling SQL Server, and it's as easy to write scripts or tools that target multiple servers as it is to target a single server.



SMO replaces SQL Distributed Management Objects (SQL-DMO) from SQL Server 2000, and is designed primarily for use with the .NET Framework.

Using SMO isn't for the faint of heart, and a complete discussion of its capabilities is beyond the scope of this book; consult the SQL Server 2005 Books Online, or Microsoft's MSDN Library, for a complete reference to SMO as well as examples. Briefly, however, SMO is a set of managed classes that are accessible to the .NET Framework (VB.NET, for example) languages, and which expose management functionality for SQL Server 2005. For example, the following VB.NET snippet uses SMO to initiate a backup of the AdventureWorks database, backing it up to a file named C:\SMOTest.bak:

```
Imports Microsoft.SqlServer.Management.Smo
Module SMOTest
    Sub Main()
        Dim svr As Server = New Server()
        Dim bkp As Backup = New Backup()
        bkp.Action = BackupActionType.Database
        bkp.Database = "AdventureWorks"
        bkp.DeviceType = DeviceType.File
        bkp.Devices.Add("c:\SMOTest.bak")
        bkp.SqlBackup(svr)
    End Sub
End Module
```

 For the original text of this example, as well as a C# example and a longer discussion on using SMO, visit <http://www.sqldbatips.com/showarticle.asp?ID=37>.

SMO isn't accessible exclusively from .NET; it's available to Component Object Model (COM) based languages, as well, through .NET's COM interoperability interfaces. For example, here's a VBScript version of the previous example:

```

Const BackupActionType_Database = 0
Const DeviceType_File = 2

Set svr = CreateObject("SQLSMO.Server")
Set bkp = CreateObject("SQLSMO.Backup")

    bkp.Action = BackupActionType_Database
    bkp.Database = "AdventureWorks"
    bkp.DeviceType = DeviceType_File
    bkp.Devices.Add("c:\SMOTest.bak")
    bkp.SqlBackup(svr)

Set bkp = Nothing
Set svr = Nothing

```

As you can see, using SMO is fairly straightforward: Obtain a reference to the appropriate objects (SQLSMO.Server and SQLSMO.Backup, in this case), and provide the appropriate property settings to do what you want. Then simply call a method (SqlBackup) to perform whatever management action you want. Expanding this to run against multiple servers is also straightforward: Read server names from a text file (for example), and simply create a loop that runs the script for each computer name in the file.

In summary, although commercial solutions specifically designed for SQL Server scale-out management aren't widely available, SQL Server has plenty of functionality built in to make things easier. You can create SQL Server Agent jobs that target the servers in your scale-out solution, or write scripts that perform whatever management tasks you need against whatever servers are appropriate.

## Hardware Scenarios for Easier Scale-Out Management

There are times when your hardware selections can make a major impact on the administration of your scale-out solution. Although most architects' first approach for a scale-out solution is massive rack mount servers and external drive arrays, a more careful examination of what's available, as well as the advantages and disadvantages of various hardware solutions, can result in an equally functional and well-performing scale-out solution that's actually easier to manage.

## Blade Computing

One of the biggest problems with a scale-out solution is hardware maintenance and the sheer space required by large rack mount servers. Blade servers, such as the Dell PowerEdge 1855 or the HP ProLiant BL series, can help with these problems.

Blade computing begins with a *chassis*, which provides power, cooling, keyboard-mouse-monitor connectivity, and other shared services. The actual *blades* are essentially a super-motherboard, containing all the core elements of a server: Processor, memory, and typically some form of local storage. The blades fit within the chassis, and function as entirely independent servers: Each blade has its own network connectivity, its own operating system installation, and so forth. However, because each server—blade, that is—lacks an independent chassis, power supply, cooling system, and so forth, it's much smaller. In all, blade computing can usually fit about 50% more computing into the same space as traditional rack mount servers.

Blade computing is often bundled with centralized systems management software, which allows single-seat administration of the entire chassis (monitoring for power, cooling, and other infrastructure services), as well as software for managing the blades themselves (that software often provides agents for multiple operating systems), such as providing remote control, installation, monitoring, and overall management.

Just because blades are smaller than traditional servers doesn't mean they're less powerful. In fact, blade servers are often equipped with the same high-end processors you might find in any other server suitable for a scale-out scenario: Fast x64 processors (such as the Intel Xeon series or the AMD Opteron series), 32GB of RAM (depending on the blade model), a local 15,000RPM SCSI hard drive (often attached directly to the blade), and other high-end features. Daughter cards—the blade equivalent of a PCI expansion card—provide Fibre Channel connectivity, gigabit Ethernet, and other advanced functions. While massive scale-up is not possible within a blade—there are no 64-way blade computers, for example—the very purpose of scale-out is to spread workload across multiple servers, and blade computing makes that easier to do while helping to reduce overall administrative overhead as well as data center real estate.

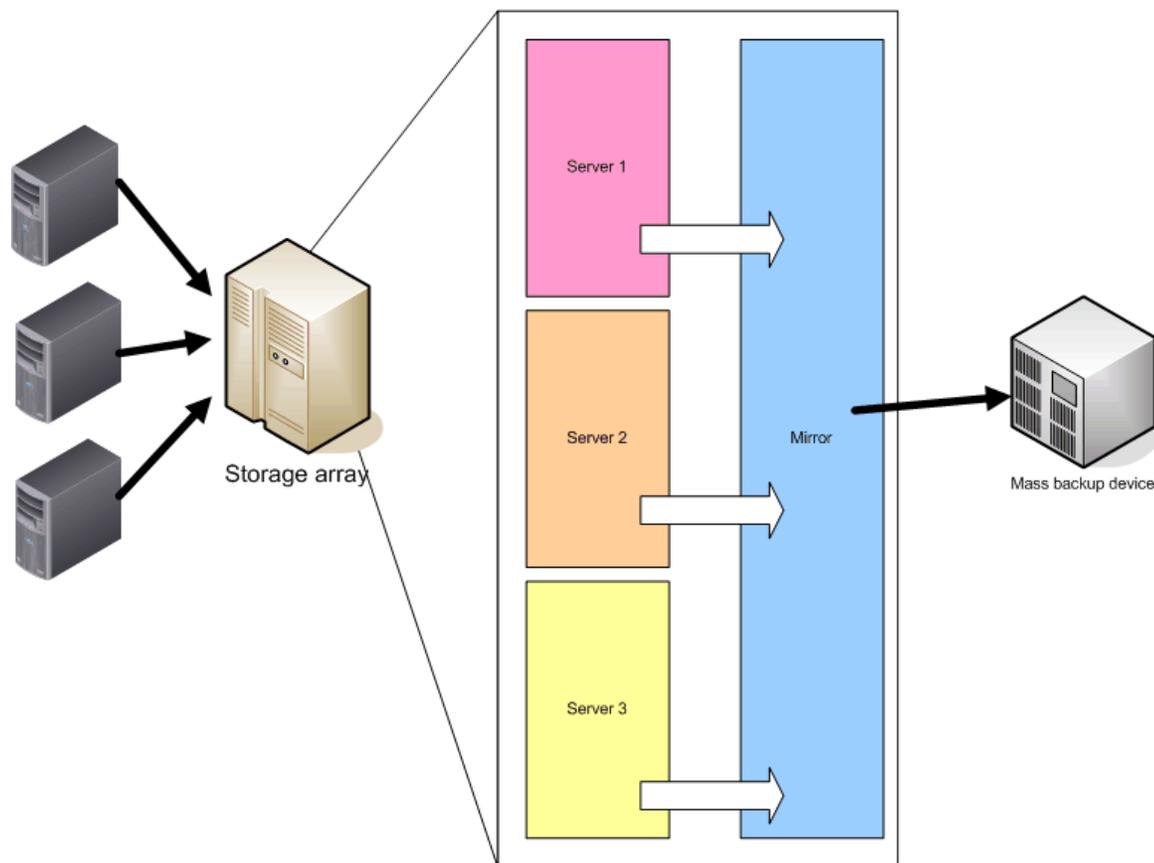
## Storage Solutions

I mentioned before that storage solutions can provide answers to some major maintenance and management problems in scale-out solutions, particularly data backup. And many scale-out solutions do rely heavily on high-end storage solutions to make tasks like data backup easier, and to make it possible to get a single, consistent backup of the entire application's data set.

Figure 7.13 illustrates the basic concept. A single external storage system—likely a SAN—provides partitions for three SQL Server computers. The storage system uses its own functionality to mirror all three partitions to a fourth area, which isn't directly accessible to any of the servers. This feature is fairly common in high-end storage systems, and can be used (for example) as a form of fault tolerance (whether the three server-accessible partitions are mirrored to one large partition or each to their own individual mirror is an implementation detail that differs depending on the storage solution in use and the goals of the mirroring). In this example, the mirror can be periodically broken, making it a point-in-time snapshot of the application's overall data store. The servers continue using their own accessible partitions, but the mirror is used as the source for a backup operation, which can take however long it needs to write the data to tape, magneto-optical storage, or whatever medium is appropriate. Once the backup operation

is complete, the mirror is restored, and the storage system brings it up-to-date with the three servers' partitions.

 If both backup and fault tolerance capabilities are desired, then two mirrors might be used: Mirror set 1 would provide fault tolerance for the servers' partitions, and would never be broken except in case of a disaster; mirror 2 would be periodically broken and then restored, and would be used by the backup solution.



**Figure 7.13: Using storage to improve management and maintenance.**

This technique is one way in which creative use of a high-end storage system can help solve otherwise tricky management problems in a scale-out solution. By leveraging the storage solution's own capabilities for mirroring data, both fault tolerance *and* a large-scale backup solution can be put into place. SQL Server's own backup capabilities wouldn't be required, since the backup would be taking place entirely behind the scenes, without impacting SQL Server in any way.

## Summary

In this chapter, I've covered some of the biggest challenges facing administrators in a scale-out solution, including challenges related to maintenance, management, and monitoring. Since scale-out solutions by definition involve multiple servers, and since much of the world's SQL Server management practices are single-server oriented, you do need to exercise some creativity in researching and selection techniques and solutions to reduce the overhead of managing multiple servers. It's entirely possible, though, as I've pointed out in this chapter, to minimize the additional administrative overhead imposed by having multiple SQL Server computers in a solution. By using SQL Server's native features, commercial solutions, and by rolling your own solutions when necessary, scale-out administration can be made nearly as straightforward as single-server administration.

## Content Central

[Content Central](#) is your complete source for IT learning. Whether you need the most current information for managing your Windows enterprise, implementing security measures on your network, learning about new development tools for Windows and Linux, or deploying new enterprise software solutions, [Content Central](#) offers the latest instruction on the topics that are most important to the IT professional. Browse our extensive collection of eBooks and video guides and start building your own personal IT library today!

## Download Additional eBooks!

If you found this eBook to be informative, then please visit Content Central and download other eBooks on this topic. If you are not already a registered user of Content Central, please take a moment to register in order to gain free access to other great IT eBooks and video guides. Please visit: <http://www.realtimepublishers.com/contentcentral/>.