ONE

Overview of SQL Server 2000

SQL Server is a relational database engine that supports both very small databases as well as enterprise class databases, all in a single package. There are many challenges in managing database systems, in particular in the area of performance.

Since the introduction of V7, SQL Server has introduced several new and improved capabilities that enhance its performance. The goal of this chapter is to outline the important improvements and lay the foundation for the rest of the book.

In general, SQL Server 2000 has been improved on in several areas: ease of use, scalability, reliability and performance.

It is important to understand the components within SQL Server that consume resources. Then a framework can be developed to understand how to improve the performance of the resource consumers.

Enhancements to SQL Server 2000

There have been several improvements to SQL Server 2000, especially in the areas of scalability and extensibility.

Large Memory Support

One of the improvements for Windows 2000 server environments is support for AWE, the Address Windowing Extensions API. AWE enables Windows

2000 Advanced Server and DataCenter Server to support more than 4 GB of physical memory, supporting up to 64 GB of physical memory using Intel's ESMA. Windows 2000 Advanced Server can support up to 8 GB of physical memory and DataCenter Server can support up to 64 GB of physical memory.

Windows 2000 Advanced Server and Windows 2000 DataCenter server both support Intel's PAE architecture. PAE stands for Physical Address Extensions and is Intel's implementation to extend memory support beyond the 32-bit physical barrier. Windows NT V4 supports greater memory support through the use of Intel's PSE36 driver. This driver interfaces directly into the processor's PSE 36-bit addressing mode to manage use of memory above 4 GB. PAE natively provides better support and performance for 4-KB pages than does PSE36, so PAE was chosen to be used with Windows 2000. PSE36 driver was not converted to Windows 2000.

The Enterprise Edition of SQL Server for Windows 2000 Advanced Server and DataCenter support the AWE API for expanded memory support. AWE is a programming interface that enables applications to address large address spaces. Memory is acquired as nonpaged, which is mapped as a view to the 32-bit address space.

Internet Support

Microsoft SQL Server 2000 supports Internet access and provides a foundation for applications that use Internet technology. SQL Server 2000 integrates with several other products to extend the functionality of Internet applications. It can integrate with Windows 2000 to provide security and encryption services to supply a secure data store. It can integrate with Microsoft Site Server to support and provide scalable e-commerce solutions.

Internet applications can access data stored on SQL Server databases in a couple of different ways. SQL Server 2000 supports the access through the use of URLs. Data can be returned to Internet applications either as standard rowsets, or, through the use of the FOR XML clause, it can return the data as an XML document. URL support includes direct access to database objects through standard TSQL queries issued as part of the URL. It also supports executing XML document templates, which consist of SQL statements. As the template is selected, the statements within the template are executed. The URL can also be used to execute Xpath queries.

Data can also be exported from SQL Server databases to be posted to Internet servers. Data can be posted to HTTP and FTP locations. This is one of the key improvements for replication in SQL Server 2000. Replicated data can be posted and retrieved from an FTP site. Data can also be exported out of SQL Server as HTML pages. SQL Server 2000 also supports Web pages using IDC, the Internet Data Connector, as well as ASP, Active Server Pages.

SQL Server Federations

Windows DNA defines the framework for building application infrastructures to support large data access needs. SQL Server is presented the opportunity to scale to support large environments through its integration with the Windows DNA framework using COM. Windows DNA, or Windows Distributed Internet Applications Architecture, serves as the foundation for the architecture of SQL Server federations. The Windows DNA is an application development model, which defines how to design and develop distributed applications to support existing data structures. It also provides the extensibility to support both Internet application interfaces as well as traditional client/server interfaces. The design of Windows DNA allows it to be flexible to scale to meet the challenges of large-scale data requirements as well as embracing newer technologies. Windows DNA addresses the issues of integrating client/server applications with Internet technologies.

A federation of SQL Server systems is created by partitioning database resources. Specifically, tables from one or more databases are broken down into smaller tables and are logically spread across multiple servers. Each of the servers in the federation shares the responsibility to cooperate with the others to present a single representation of the data to user interfaces.

Key to the functionality of partitioned views is the ability to support updateable views. Updateable views allow applications written on SQL Server to scale to support very large database applications, such as Web applications. The query optimizer in SQL Server 2000 has been enhanced to support techniques to limit the amount of distributed processing required in optimizing queries that use partitioned views. The types of applications that are best used for federated SQL Servers are large-scale OLTP and updateable Web applications.

Multiple Instances

SQL Server 2000 introduces the ability to host multiple implementations of SQL Server on a single system simultaneously. Multiple instances support more than one instance of the SQL Server database engine including its own set of system and user databases. These instances are not configured to share resources or services. This allows for great flexibility and availability of data. Each instance has its own copy of Microsoft SQL Server and SQL Server Agent services. Administrators can configure multiple server-wide settings on one instance, and support a different set of system options on another instance.

Backup and Restore Functions

Backup and restore functions provide the mechanism for data availability by maintaining a copy of data on disk, tape or both. There have been several

advancements for the backup and restore functions such as support for up to 64 tape devices in a single backup or restore operation. There are three recovery models that work to balance the speed of recovery operations to that of performance in the production environment. Each of the recovery models works best with a combination of full and differential backups. The full recovery and bulk-logged recovery models work best when using Transaction Log backups as well.

SQL Server supports three types of backup operations: full backup, differential backup and transaction log backup. Differential backups were introduced with V7. You can also select to backup an individual data file or an individual file group. If the backup window is very small and the quantity of data to backup is very large, consider a combination of multiple tape devices and a rotating schedule of full backups, differential backups, filegroup backups and Transaction Log backups.

Replication Enhancements

Probably the biggest performance improvement for replication is the option to store replication information in several different locations. You can now store compressed replication snapshots to a network location, an FTP site or even removable media. This allows you to remove the load on the network if need be. One option is to save snapshot files to an FTP directory and make them available to subscribers from there. This allows the SQL Server to offload the responsibility of transferring the replicated file. It off-loads the transfer to the FTP server.

Log Shipping

Log shipping has been built into the SQL Server database maintenance utility. It automates Transaction Log backups from one system to be automatically restored to one or more SQL Server systems. The primary system that contains the database is the source server, and the systems to which the data are restored are the destination servers. Log shipping supports multiple destination servers and provides a means for supporting multiple copies of data.

Since log shipping applies the Transaction Logs from one system to one or more destination servers, all data structures, including tables and indexes, must be the same.

Index Enhancements

SQL Server 2000 introduced the ability to index views. When an index is created on a view, the result set of the view is stored in the database. This method is best used when the data referenced by the view do not change often. It can improve the performance of queries that store aggregated data,

especially if the data doesn't change often. The query optimizer will consider indexes on views when optimizing other queries, even if they do not reference the view in the FROM clause.

Indexes can be stored in either ascending or in descending order. They can also be created on computed columns. Index creation has also been optimized to take advantage of parallel scanning and sorting operations.

Windows 2000 Security Enhancements

Running SQL Server 2000 with Windows 2000 allows you to use Kerberos to support authentication between client and server. It can also be used for impersonation—passing the credentials of one user to another system so that all processes completed on behalf of the user use the user's credentials. SQL Server 2000 with Windows 2000 uses Kerberos and delegation by default to support both authentication and SQL Server logins.

Full-Text Search

Full-text search was introduced in V7 for character-based data. Indexes can be created built on keywords in defined columns. SQL Server implements extensions to support full-text indexes for proximity searches.

Full-text searching has been enhanced in V2000 to include support for image filter and change tracking. The change tracking allows you to log all the changes to the data and then control when updates should be performed. Image filtering is also supported to allow you to index and query data stored in image columns.

SQL Server V7.X Enhancements

In addition to the enhancements in V2000, SQL Server V7 introduced several key features that laid the foundation for the current version. SQL Server V7 defined a lot of the architecture from which SQL Server 2000 evolved.

Simplified Configuration

One of the challenges in managing previous versions of SQL Server was the number of configuration options that needed to be manually set and tuned for. SQL Server V7 significantly reduced the amount of manual configuration. For servers with large amounts of data storage, the only configuration option that on occasion needed to be modified was max asynch I/O.

Dynamic Space Management

As of V7, SQL Server databases are no longer stored in devices and segments; rather, they are stored in operating system files. This not only simplifies management, it also offers several residual benefits. Database files can grow and shrink on demand, as can other operating system files. Databases are minimally made up of two files, one for data and other objects, and another for the log file. Optionally, databases can consist of multiple files that can spread out the I/O load.

Data Storage

Database pages have increased from 2-KB pages to 8-KB pages in V7. Extents have increased from 16 KB, 8 2-KB pages to 64 KB, 8 8-KB pages. The maximum number of bytes in a row is up to 8,060. Tables can support up to 1,024 columns significantly up from 250 columns. These enhancements have resulted in better performance through more efficient I/O, a decrease of page splits and a reduction in the overhead on pages.

Parallel Query Execution

SQL Server V7 introduced the ability to split a query into multiple tasks and allow each task to be executed on a separate processor. CPU-bound queries that examine large quantities of rows are best suited for parallel execution. SQL Server manages the level of parallelism based on hardware resources and query optimization. If hardware resources are inadequate, the degree of parallelism will be decreased. A parallel query is more costly to resources than is a query that is executed simultaneously.

Cost-Based Locking

Row-level locking is supported for both data rows and indexes. The lock manager was optimized in V7 to complete requests faster with less synchronization. Lock intents can be placed on a higher level in the hierarchy so that other lock requests do not need to search the entire hierarchy to determine what locks it can place on an object. The lock manager is self-tuning in terms of the resources it uses.

Improved Execution of Stored Procedures

In previous versions of SQL Server, each use of a stored procedure by a user required a separate copy to be placed in the procedure cache for that use. Since SQL Server V7, one copy of the compiled procedure is used from the procedure cache for each user connection. SQLPrepare and |CommandPre-

pare interfaces can share plans. Ad hoc plans can also be shared by passing parameter markers through to the database. Stored procedures also provide support for referencing objects before they have been created.

Hash and Merge Joins

SQL Server V7 included support for hash and merge joins along with nested loop joins. The query optimizer will determine which to use for processing complex queries. Star schema joins are also supported. This is a vast improvement over V6.5, which only supported nested loop joins.

Performance Analysis Tools

There are several performance analysis tools for SQL Server and Windows NT/2000. Windows NT/2000 Performance Monitor is integrated with SQL Server. From either performance monitor or SQL Server tools, the impact that SQL Server puts on the server system can easily be measured. Performance Monitor includes the SQL Server objects and counters that are useful in monitoring the performance of SQL Server.

SQL Server Profiler, a replacement for Trace, is a workload capture and measurement facility that can provide information about how a particular workload is performing. It can measure, for instance, the duration of queries to assist in determining which queries cause the most drain on resources. The workload files generated from it can be used as input to the Index Tuning Wizard. The workload file can also be replayed using SQL Server Profiler to allow you to step through actions as they occurred on your server.

The Index Tuning Wizard—as its name implies—makes index recommendations. It makes its index recommendations by analyzing a workload. A workload can be a file or table created by SQL Server profiler or an SQL script. The Index Tuning Wizard can only make recommendations based on the queries presented in the workload, so it is critical for the workload to be representative of the work you are expecting the system to be used for.

Query Analyzer is a robust graphical interface providing access to several management and tuning functions for SQL Server. Query Analyzer allows administrators to write and execute queries, perform system maintenance such as backup, and restore options, as well as analyze the execution plan of queries and tune indexes and statistics. New to SQL Server 2000 is the object browser. As its name implies it offers a hierarchical browser of all the objects in each database, including tables, views, indexes and constraints.

Resource Consumers

There are several consumers of resources for SQL Server, both to support users as well as to manage the system itself. The goals of this book include understanding what the issues are, learning what is using the resources and knowing how to tune for it. Resources are consumed by components such as the data store (i.e., the Storage Engine) and the execution engine (i.e., the Query Processor). The Storage Engine manages how data is located on disk and how it is managed. The Query Processor is responsible for managing query optimization and query execution. There are several internal SQL Server components that are critical to the success of the Storage Engine and the Query Processor.

Some of the more important internal resource consumers are LazyWriter, Checkpoint, worker threads, read ahead manager and the log manager. Each of these plays a key role in SQL Server and their actions need to be monitored and managed for optimal performance.

LazyWriter

SQL Server uses buffer pages to store data modifications. It also needs to be able to synchronize memory pages that have been updated with data that is stored on disk. That is where LazyWriter comes into play. The purpose of the LazyWriter is to flush dirty cache pages, keeping a supply of available free cache pages. Free buffers are 8-KB data cache pages that are free of data. As SQL Server writes each cache buffer out to disk, it initializes the identity of the cache page so that other data may be written into the "new" free buffer.

SQL Server automatically manages and adjusts itself to have available an acceptable number of free buffers. V7 required managing the max asynch I/O configuration option to modify the behavior of LazyWriter disk I/O; SQL Server 2000 handles this dynamically. The max asynch I/O option controlled the number of 8-KB disk write requests that SQL Server can simultaneously submit to the disk I/O subsystem. To determine the appropriate level of max asynch I/O, measure the disk queuing counters for the disks. If SQL Server cannot maintain an acceptable level of free buffers, adding additional physical disks will spread out the load and help decrease the amount of I/O per disk.

Checkpoint

Checkpoints are used to minimize the amount of information that needs to be restored from the log file on recovery. A checkpoint is a determination of the amount of recovery time in the event of failure. If the recovery time exceeds the recovery interval—a threshold for the maximum amount of time to perform a recovery—then the system will execute a checkpoint. The

recovery interval is self-tuning with a goal to maintain recovery times of less than one minute. The recovery interval determines the maximum number of records that can be processed within a specified period of time. Checkpoint ensures that completed transactions are written from the disk cache to the disk regularly.

Checkpoint writes dirty pages that were marked as dirty at the last checkpoint to the data files. The buffer cache pages that have been modified are the dirty pages. Data is maintained in the buffer even after it has been written to disk by Checkpoint, allowing other requests for the data to be referenced in the buffer pages. Once LazyWriter has flushed the data, it is no longer contained in the buffer pages.

The goal is to allow LazyWriter and worker threads to manage the process of writing dirty pages. Checkpoint adds an extra wait time prior to writing data, to allow LazyWriter and worker threads more time to write the information themselves. To make Checkpoint more efficient with a large number of pages needing to be flushed is to sort the pages prior to flushing to disk. This can minimize disk seek and can potentially take advantage of sequential disk I/O.

Checkpoints occur:

1. when database options are changed
2. manually by an administrator
3. at system or SQL Server shutdown
4. automatically for autorecovery

If SQL Server is configured for simple recovery mode, checkpoints will occur:

1. when the log files becomes 70% full
2. automatically for autorecovery

Worker Threads

Worker threads are a pool of Windows threads that are used to service batches of SQL commands. Specifically, they service user requests, network connections and checkpoints. Worker threads take on most of the write activity in writing dirty cache buffers to disk.

The default value for worker threads is 255. That is usually sufficient to handle an average amount of incoming batch requests. Each user request, network connection and checkpoint is allocated its own worker thread up to a maximum value for worker threads. If the usage demands for worker threads exceed the maximum value, the requestors will share the worker threads. The maximum value for worker threads is 1024. All allocated worker threads, whether in use or not, consume resources.

Log Manager

The Log Manager is responsible for verifying that all write operations are complete and, on system failure, the integrity of the data in the database is sound. The Transaction Log is used to control database modifications and to provide a method for orderly recovery following unexpected system or database downtime. To ensure recoverability, the Log Manager waits to receive a signal from the disk subsystem that the changes have been written to the associated log file. It is therefore important to have sufficient I/O capacity for the load imposed by writing the transaction information to the log. Once data has been written to the Transaction Log, it can be written to disk. At checkpoint, the contents of the Transaction Log are synchronized with the actual data pages. After checkpoint, the data in the tables reflect all committed transactions.

For write operations, the Transaction Log is one of the most heavily accessed data structures, since each operation is logged. For instance, for each modification of data, both the old and new data is written to the Transaction Log. For performance and recoverability purposes, it is recommended to separate data files from Transaction Log files. Spreading the load out onto multiple physical disks will aid in performance, while mirroring or striping the disks will assist in recoverability and, to some degree, performance.

Read-Ahead Manager

The Read-Ahead Manager is responsible for managing parallel data scan operations, known as read-ahead. When certain query operations, such as table scans and large sequential searches, are detected by the Query Processor, it allocates a thread to read-ahead data in the background. Therefore once the request for the data is actually made, the data is already available. The Read-Ahead Manager is self-configuring and self-tuning by virtue of the Query Processor detecting situations where performance can be increased by read-ahead. Read-Ahead activities are capable of 64-KB I/O.

Further, the Read-Ahead Manager utilizes the Index Allocation Map as its storage structure. The Index Allocation Map, more commonly referred to as the IAM, records the location of extents in a bitmap into an 8-KB page. Since the IAM is compacted through its use of a bitmap, frequently used IAM pages tend to remain in the buffer cache.

Too often, read-ahead can actually degrade performance since it can fill cache pages with data that are not needed. It is important to tune queries so that a minimum number of pages are actually brought into cache to satisfy the request. Help the query optimizer make intelligent decisions about what to bring into cache by ensuring current, usable and appropriate indexes and statistics are available.

Key Considerations for Tuning SQL Server

SQL Server is mostly self-tuning and self-configuring, yet still requires monitoring to ensure it is tuned to meet the expectations of the organization. The best way to work with SQL Server—to allow it to make most of the tuning and optimization decisions—is to monitor and analyze the behavior of your environment.

Monitor the allocation and use of physical resources. A system that is low on physical memory will produce a significant number of bottlenecks, evident both as memory bottlenecks and also disk bottlenecks. Inappropriately tuned queries can lead to significant disk and processor bottlenecks, by producing hot spots or queries that time out.

Windows 2000 and SQL Server provide several tools that are instrumental in managing and monitoring both physical resources, such as memory and CPU, as well as application resources, such as indexes and statistics. Performance can be monitored in Windows NT/2000 by using Performance Monitor, Event Viewer, Task Manager, and Network Monitor, as well as other tools. SQL Server can be tuned and monitored through the use of Performance Monitor, Alerts, SQL Server Profiler, ShowPlan and the Index Tuning Wizard.

Why Are Tuning and Optimization Necessary?

If SQL Server is self-tuning and self-configuring, why do I need to tune it? SQL Server makes a best attempt to tune itself, but is often at the mercy of other variables. For instance, disks that are inefficient or are failing may display themselves to SQL Server as a slow-performing query. It is important to monitor SQL Server and add in the human factor of being able to consider all aspects of the environment. While performance problems can and do exist in SQL Server environments, they are not limited to just the application.

Monitoring is useful in detecting potential hardware problems. It is also useful in capacity planning. By understanding the current consumption of resources and how the allocation of resources has changed over time, an administrator can extrapolate information to predict what resources she will need in the future.

As the use of the database and the contents of the database change over time, the optimization of the database should also be reviewed. If users are no longer querying for information on particular tables or on columns within tables, indexes will need to be re-created to meet current needs. Indexes that are not useful degrade performance. If the use of the database has changed, evaluate the indexes to determine if they meet the needs of the workload.

Server Profiler and the Index Tuning Wizard can assist in measuring and analyzing a workload.

Summary

The majority of the new features introduced for SQL Server 2000, including SQL Server Federations, internet support, multiple instances and log shipping, are positioning SQL Server to be an enterprise class solution. Along with benefits offered by advances in the underlying operating system such as large memory support and support for Kerberos security, SQL Server can now offer the scalability and security needed by today's larger database environments.

Many of the features of SQL Server also impact performance. Database files are now stored in operating system files. This simplifies management, allowing databases to grow and shrink on demand, although a database that grows frequently will degrade performance. Database management is also simplified by the ability to distribute database files onto multiple disks to improve performance. Row-level locking is supported for data and index pages. Row-level locking provides a finer granularity to locking, thereby increasing the performance of concurrent access.

Some of the main resource consumers for SQL Server are LazyWriter, Checkpoint and worker threads, along with the log manager and read-ahead. Each of these components are responsible for tracking data, ensuring data is successfully written to both log file and disk as well as reading information into cache to satisfy requests in a timely manner.

SQL Server is mostly self-tuning and self-configuring, but as with all applications, it needs to be monitored for capacity and potential problems. Unless a database is read-only and its use is static, databases also need to be analyzed to optimize their current use. As the data in the database change over time, so may the use of the database.