

Best Practices for Microsoft® SQL Server on Hitachi Universal Storage Platform® VM

For Online Transaction Processing Databases

By Eduardo Freitas

June 2009



Summary

The Hitachi Universal Storage Platform® VM brings enterprise-class availability, performance and ease of management to organizations of all sizes for the increasing number of business-critical applications. It is not only an ideal platform for Microsoft SQL Server deployment but also easy to configure and deploy with Microsoft Exchange and other common applications on Windows Server 2008.

The Universal Storage Platform VM is robust due to Hitachi Dynamic Provisioning software, with features such as thin provisioning and wide striping, which are radical changes from the traditional method of configuring and presenting storage for use in a SQL environment. With Hitachi Dynamic Provisioning, customers can minimize the complexity and deployment time of their environments even when running many applications such as SQL Server databases that require storage provisioning and varying levels of capacity and performance.

The information in this paper enables storage and database administrators to successfully plan, deploy and monitor SQL Server environments by providing them with a good understanding of the interaction between the many components of SQL Server and Hitachi Dynamic Provisioning software on the Universal Storage Platform VM.



Contributors

The information included in this document represents the expertise, feedback and suggestions of a number of skilled practitioners. The author recognizes and sincerely thanks the following contributors and reviewers of this document:

- Rick Andersen, SCSSI — Solutions Engineering
- Allan Benway, Product Performance Management
- Patricia Brailey, SCSSI — Solutions Engineering
- Steve Burr, Global Solution Services
- Ron Lee, Product Performance Management
- Lisa Pampuch, Application Solutions
- Gilbert Rangel, Product Performance Management



Table of Contents

Hitachi Universal Storage Platform VM	1
Features	2
Enterprise-class Software Tools.....	2
SQL Server 2008.....	4
Interactive Administration	4
Database Flexibility	4
Business Continuity.....	5
Security Enhancements	5
Database Layouts.....	6
Database File Considerations	7
RAID Level Considerations	8
System Design Considerations for Hitachi Dynamic Provisioning Software.....	10
Configuration Options	12
Performance, Availability, Capacity Utilization and Ease of Management Capabilities	15
Host Considerations	16
Performance Monitoring	19
Windows Performance Monitor	19
Hitachi Performance Manager Feature	19
Hitachi Tuning Manager	20
SQL Server Data Management Views and Functions	21
SQL Server 2008 Management Data Warehouse.....	22
References	23
Appendix A: Using Performance Monitoring Counters	24
Appendix B: DMV and DMF Examples.....	29
Monitoring Database File Information	29
Monitoring tempdb Usage	29
Monitoring Cache Usage.....	30
Monitoring Latch Waits.....	30
Monitoring I/O Writes and Sizes.....	31
Monitoring I/O Pending	32



Best Practices for Microsoft® SQL Server on Hitachi Universal Storage Platform® VM

For Online Transaction Processing Databases

By Eduardo Freitas

The Hitachi Universal Storage Platform® VM brings enterprise-class availability, performance and ease of management to organizations of all sizes for the increasing number of business-critical applications. It is not only an ideal platform for Microsoft® SQL Server deployment but also easy to configure and deploy with Microsoft Exchange and other common applications on Windows Server 2008.

The Universal Storage Platform VM is robust due to Hitachi Dynamic Provisioning software, with features such as thin provisioning and wide striping, which are radical changes from the traditional method of configuring and presenting storage for use in a SQL environment. These features enable storage and database administrators (DBAs) to deploy SQL Server environments on the Universal Storage Platform VM while achieving these goals:

- Reducing deployment time
- Increasing capacity utilization
- Consolidation of resources
- Increasing overall database performance
- Leveraging thin provisioning

With Hitachi Dynamic Provisioning software, customers can minimize the complexity and deployment time of their environments even when running many applications such as SQL Server databases that require storage provisioning and varying levels of capacity and performance.


The information in this paper enables storage and database administrators to successfully plan, deploy and monitor SQL Server environments by providing them with a good understanding of the interaction between the many components of SQL Server and Hitachi Dynamic Provisioning software on the Universal Storage Platform VM.

Hitachi Universal Storage Platform VM

The Universal Storage Platform VM blends enterprise-class functionality with a smaller footprint to meet the business needs of entry-level enterprises and fast-growing mid-sized organizations, while supporting distributed or departmental applications in large enterprises.

Now, smaller organizations can enjoy the same benefits as large enterprises in deploying and managing their storage infrastructure in a way never possible before. The Universal Storage Platform VM, powered by the Hitachi Universal Star Network crossbar switch architecture, delivers proven and innovative controller-based virtualization, logical partitioning and universal replication for open systems and mainframe environments in a rack-mounted storage services platform.

The industry's highest reliability and availability storage solution is backed by a set of storage and data services that include thin provisioning with Hitachi Dynamic Provisioning software, application centric storage



management and logical partitioning, as well as simplified, unified data replication across heterogeneous storage systems.

The Hitachi Universal Storage Platform VM packages and delivers critical services like these:

- Virtualization of storage from Hitachi and other vendors into one pool
- Thin provisioning through Hitachi Dynamic Provisioning for nondisruptive volume expansion
- Security services, business continuity services and content management services
- Load balancing to improve application performance
- Nondisruptive dynamic data migration from Hitachi and other storage systems

Features

These features of the Hitachi Universal Storage Platform VM let you deploy applications within a new framework, fully leverage and add value to current investments and more closely align IT to your business objectives:

- Excellent performance with 1.2 million I/Os per second (IOPS), 13.3 GB/sec aggregate internal bandwidth
- Superior scalability providing up to 96PB of total storage capacity under management, including up to 94.5TB of Fibre Channel internal storage or 236.3TB of SATA internal storage
- Up to eight virtual storage machines with logical partitions, dedicating cache, ports and internally and externally attached capacity to ensure application quality of service (QoS)
- Superior multiplatform connectivity with up to 48 Fibre Channel, 24 IBM® ESCON® and 24 IBM FICON® host ports and network attached storage (NAS)
- Virtualization of internally and externally attached storage from Hitachi, EMC, IBM, Sun, HP and other manufacturers
- Hitachi Dynamic Provisioning Software for thin provisioning of internal and virtualized external storage
- Nondisruptive movement, copy and migration of data between storage, including other vendors, without interrupting the application
- Universal replication across heterogeneous platforms for open and mainframe environments
- Active workload balancing and tuning
- Multi-protocol consolidation with Fibre Channel, IBM® FICON® and ESCON® and NAS
- Single-pane-of-glass management with Hitachi Storage Command Suite software
- Powered by the fourth generation Hitachi Universal Star Network crossbar switch architecture

Enterprise-class Software Tools

Several tools and software applications make managing and monitoring a SQL Server 2008 environment easier. Hitachi offers several enterprise class management applications that allow administrators to easily provision storage with automated wizards and robust command-line interfaces to monitor performance and capacity and to ensure availability.

Hitachi Basic Operating System Software

Hitachi Basic Operating System software is an integrated set of automated tools that simplify storage management, partitioning and virtualization capabilities of the Hitachi Universal Storage Platform VM. It includes the following components:

- **Hitachi Storage Navigator** — Monitors and manages Hitachi Enterprise storage systems through a GUI accessible through an Internet browser or by using a command-line interface. Use Storage Navigator to create RAID groups and logical units and to assign those logical units to hosts. Storage Navigator is also useful to monitor events and status on the various components that comprise the Universal Storage Platform

VM and for setting up advanced features such as dynamic provisioning, replication, storage virtualization and security.

- **Performance Manager Feature** — Acquires information on the performance of RAID groups, logical units and other elements of the storage system while tracking the utilization rates of resources such as hard disk drives and processors. Information is displayed with line graphs in the Performance Manager window of Storage Navigator or saved in .csv files that can later be analyzed by the SQL or storage administrator.
- **Hitachi Device Manager** — Provides centralized management of all Hitachi storage systems, including the Universal Storage Platform VM. Device Manager can link to Storage Navigator and has the ability to provision using storage pools, manage replication between storage systems and logically group resources for more efficient management. It also provides the device interface into Hitachi storage systems for the heterogeneous Hitachi Storage Services Manager software and other third-party storage management tools. Device Manager's Provisioning Assistant is a value-added component that enables administrators to integrate and manage various models and types of storage systems as a single, virtual storage pool, allowing storage administrators to do more with less. These are some of Device Manager's key features:
 - Central consolidation and management of multiple, local or remote Hitachi storage systems, all from a single point of control
 - Operating system independent, holistic and granular view of all storage
 - Central repository of configuration, utilization, capacity and status of all managed storage
 - Open API based on industry standards common interface management (CIM) and Storage Management Initiative Specification (SMI-S)
 - Enhanced user authority control feature

Hitachi Dynamic Provisioning

Hitachi Dynamic Provisioning is an advanced provisioning software product and is available on the Universal Storage Platform V and VM. It allows storage to be allocated on an as-needed basis to applications such as SQL Server 2008. With Hitachi Dynamic Provisioning software, an organization's overall storage utilization improves while all hosted applications are performance tuned (via I/O workload balancing) for operational efficiency. Compatible with the advanced features and reliability of the best-in-class Hitachi Universal Storage Platform V and Hitachi Universal Storage Platform VM storage systems along with all Hitachi Services Oriented Storage Solutions, Hitachi Dynamic Provisioning software offers reduced capital and operating costs for improved return on storage investment.

Hitachi Dynamic Link Manager Advanced

Hitachi Dynamic Link Manager Advanced is a software package that consists of multipathing software and Hitachi Global Link Manager software. Hitachi Dynamic Link Manager, which is installed on each SQL Server host, includes capabilities such as path failover and failback, and automatic load balancing to provide higher data availability and accessibility. The Hitachi device-specific module is integrated into Hitachi Dynamic Link Manager and can effectively replace the generic Microsoft MPIO device-specific module that is included in Windows Server 2008.

Hitachi Tuning Manager

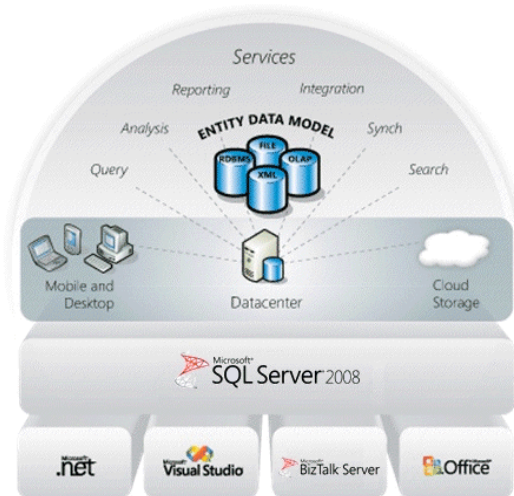
Hitachi Tuning Manager software enables you to proactively monitor, manage and plan the performance and capacity for the Hitachi storage that is utilized by your SQL Server environment. Hitachi Tuning Manager software consolidates the statistical performance data from the entire storage path. It collects performance and capacity data from the operating system, SQL Server databases, switch ports, storage ports on the storage system, RAID groups and the logical units where the SQL Server files are located. It also provides the administrator a complete performance picture. It can provide historical, current and forecast views of these metrics.

Fully integrated with the Hitachi Storage Command Suite and compliant with industry standards, Tuning Manager software improves storage performance and capacity management to maximize storage utilization on the Universal Storage Platform VM as well as all other Hitachi storage systems.

SQL Server 2008

With the release of Microsoft SQL Server 2008, Microsoft facilitates the management of any data, any place and any time. Its new features give continuity to the success and reliability of SQL Server 2005. Together with the Universal Storage Platform VM, SQL Server 2008 provides a scalable, high-performance database engine for mission-critical applications that require the highest levels of availability and security, while reducing the total cost of ownership through enhanced enterprise-class manageability for OLTP deployments. Figure 1 from Microsoft illustrates SQL Server 2008's integration capabilities.

Figure 1. SQL Server 2008 Integration Capabilities



© 2008 Microsoft Corporation. All rights reserved.

Microsoft Data Vision Platform diagram reprinted with permission from Microsoft Corporation.

Interactive Administration

Microsoft SQL Server 2008 Management Studio can manage multiple SQL Server instances from a centralized location, eliminating the need to have multiple sessions opened for each of the instances being managed. Management Studio also provides a number of standard administrative reports that can be used to view information pertaining to SQL Server instances, databases and other SQL Server objects.


Using reports generated from System Data Collect Sets, DBAs can increase their awareness of their environments by utilizing tools that enable performance monitoring, troubleshooting and tuning. Policy-Based Management simplifies the definition of configuration policies using built-in Policy-Based Management facets, Resource Governor enables you to manage SQL Server workload and resources by specifying limits on resource consumption by incoming requests.

A new, streamlined installation separates the installation of the physical bits on the hardware from the configuration of the SQL Server software, enabling organizations and software partners to provide recommended installation configurations.

Database Flexibility

New date and time data types give more flexibility and functionality to your database and provide large data ranges for use with your application. HierarchyId system type can store values that represent nodes in a hierarchy tree, enabling a more efficient way to model tree structures.

One of the most exciting new features of SQL Server 2008 is FILESTREAM, which enables the storage of data from structured, semi-structured and unstructured documents, such as images and rich media, directly within



the database. Binary large objects (BLOBs) are stored in an NTFS file system, while Win32 file system interfaces provide streaming access to the data.

The data compression feature reduces the amount of storage space needed to store tables and indexes, enabling database and storage administrators to make efficient use of the storage space available for their organizations. Page and row compression can be enabled for specific tables and indexes and compression does not require that changes be made to applications.

Other features enhance the overall experience with SQL Server and simplify the deployment of data-driven solutions and facilitate SQL developers' job:

- Integrated full-text search enables the performance of high-speed text searches on large text columns.
- Sparse columns provide an efficient way of managing empty data in your database.
- Large user-defined types (UDTs) expand the size of user-defined types by eliminating the previous 8KB limitation.
- Spatial data types enable the building of spatial capabilities in applications.

Business Continuity

SQL Server 2008 enhances the database mirroring feature that provides automatic page repair, improved performance and enhanced supportability that helps ensure database availability for your business critical applications. Automatic recovery of data pages enables the principal and mirror machines to transparently recover from 823 (operating system error or a logical I/O check failure) or 824 (logical consistency-based I/O error) types of data page errors.

Log stream compression can help maximize network throughput, especially in environments where network conditions are poor. Expect to see compression effectiveness that is similar to backup compression, with a similar impact on CPU resources.

Database mirroring requires data transmissions between the participants of the mirroring implementations. With SQL Server 2008, compression of the outgoing log stream between the participants delivers optimal performance and minimizes the network bandwidth used by database mirroring.

Security Enhancements

SQL Server 2008 database engine security enhancements include new encryption functions, transparent data encryption (TDE) and extensible key management features, clarification of DES algorithms and auditing. TDE enables the encryption of the database files without the need to alter any applications. This prevents unauthorized users from accessing a database, even if they obtain the database files or database backup files. Put together, these new features provide the highest level of security, reliability and scalability for your business critical applications.

For more information about the new features of SQL Server 2008, see the [What's New](#) page of SQL Server 2008 Books Online, the [Microsoft SQL Server 2008 Overview](#) Web site or contact your Microsoft representative.

Database Layouts

SQL Server has multiple components, each of which has a unique I/O profile based on the workload from the application server and the specific design of the application database. The three primary components are database files, transaction logs and tempdb. Table 1 from Microsoft highlights some of the different I/O patterns and sizes.

Table 1. Input/Output Patterns of SQL Server

<i>Operation</i>	<i>Random / Sequential</i>	<i>Read / Write</i>	<i>Size Range</i>
OLTP – Log	Sequential	Write	Sector Aligned up to 60K
OLTP – Log	Sequential	Read	Sector Aligned up to 120K
OLTP – Data (Index Seeks)	Random	Read	8K
OLTP – Lazy Writer	Random	Write	Any multiple of 8K up to 256K
OLTP – Checkpoint	Random	Write	Any multiple of 8K up to 256K
Read Ahead (DSS, Index/Table Scans)	Sequential	Read	Any multiple of 8K up to 256K (512K for Enterprise)
Bulk Insert	Sequential	Write	Any multiple of 8K up to 128K
Read Ahead (DSS, Index, Scans)	Sequential	Read	8K Multiples to 256K
Create Database	Sequential	Write	Up to 4MB (only log file initialized in SQL 2005)
Backup	Sequential	Read / Write	64K multiples to 4MB
Restore	Sequential	Read / Write	64K multiples to 4MB
DBCC CheckDB	Sequential	Read	8K to 64K
ALTER INDEX REBUILD (Write Phase)	Sequential	Write	Any multiple of 8K up to 128K
ALTER INDEX REBUILD (Read Phase)	Sequential	Read	Any multiple of 8K up to 256K

Note: These values may change in the future based on optimizations made to take advantage of modern storage enhancements.

Source: “Microsoft SQL Server and SAN – Lessons Learned & Best Practices” by Mike Ruthruff and Prem Mehra, SQL Server Customer Advisory Team

Database File Considerations

When deploying SQL Server, it is important to have an overall understanding of the key file types that are essential for a database. Understanding each file type's workload, along with the database type and size, enables both storage administrators and DBAs to establish storage requirements for the SQL Server environment.

SQL Server Database Files

SQL Server OLTP database file I/O is composed of random small record reads and writes. A database might include only a single database file, while those designed to support heavy transactional workloads or large schemas might use a variety of filegroup architectures to improve performance, operational convenience or availability. Acceptable database I/O response times typically range between five and 20 milliseconds. In general, whether your database maintains a single primary .mdf file or leverages multiple secondary database files (.ndf) and filegroups, the same storage design rules apply because of the random nature of the database I/O profile.

SQL Server Transaction Log File

Every SQL Server database has at least one log file that records database modifications made by each transaction. It is a critical component of the database for availability and data integrity. In the event of a system failure, the active transaction log, at a minimum, is required to bring the database back to a consistent state. The transaction logs are written before the data records are updated to the database file via the checkpoint process. The logs can be used to roll back transactions if corruption is later discovered or to recover a committed transaction if an error writing to the database occurs.

Response time and performance are critical considerations for separating the transaction log from the database files. Microsoft suggests aiming for log I/O response times between one and five milliseconds. Hitachi's optimized caching and proper storage design ensure that the logs can be written without delay.

SQL Server tempdb Files

SQL Server tempdb files are used for storage of temporary data structures and can be subject to intense and unpredictable I/O. Many best practice recommendations suggest locating tempdb files on separate RAID groups from the database and using the fastest disks available. This is generally a safe recommendation because the load on tempdb is highly dependent upon database and application design. However, if the tempdb load is well understood and monitored regularly, testing shows that it can reside on the same RAID group as the primary database without adverse effects. Accordingly, if the environment does not have sufficient physical disk I/O resources to meet the combined requirements of tempdb and the database files, performance for all databases in the SQL Server instance degrades. Therefore, you need a good understanding of your tempdb usage, regardless of where you choose to place the tempdb files.

For more information on how to determine the appropriate amount of disk space for your tempdb volumes, see the [Capacity Planning for tempdb](#) page of SQL Server 2008 Books Online.

SQL FILESTREAM Files

FILESTREAM integrates the SQL Server database engine with an NTFS file system by storing variable-length BLOB data as files on the file system. It enables the storage of structured, semi-structured and unstructured data such as images and rich media directly within the database. With FILESTREAM, DBAs can utilize transact-SQL statements to insert, update, query, search and back up FILESTREAM data from a given database.

FILESTREAM uses the NT system cache for caching file data, which helps reduce any effect that FILESTREAM data might have on database engine performance. The SQL Server buffer pool is not used; therefore, this memory is available for query processing. All this is done while maintaining transactional consistency between the structured and unstructured data at all times, enabling point-in-time recovery of FILESTREAM data using log backups. In addition, FILESTREAM works with all recovery models and all forms of backup and restore (full, differential and log).

Keep in mind that FILESTREAM data is not encrypted even when transparent data encryption is enabled. For more information about FILESTREAM overview, features, and best practices, see the [Designing and Implementing FILESTREAM Storage](#) page of SQL Server 2008 Books Online.

RAID Level Considerations

Hitachi back end directors are the most advanced in the enterprise storage industry and employ sophisticated algorithms to manage performance. The intelligent controllers provide disk interface and RAID management circuitry, offloading these tasks to dedicated embedded processors. Each Universal Storage Platform VM back end director supports RAID-1+, RAID-5 (parity RAID) and RAID-6.

All user data disks in the system are defined as part of a RAID array group of one type or another. Table 2 lists RAID types and supported disk configurations.

Table 2. RAID Types and Supported Disk Configurations

<i>RAID Type</i>	<i>Supported Disk Configurations</i>
RAID-1+	2D+2D
Mirroring	4D+4D
RAID-5	3D+1P
Distributed Parity	7D+1P
RAID-6	6D+2P
Redundant Distributed Parity	

RAID-1+

RAID-1+ is available in two data plus two data (2D+2D) and four data plus four data (4D+4D) disk configurations. The configurations include a rotating copy, where the primary and secondary stripes are toggled across the physical disk drives for performance. RAID-1+ is best suited to applications with low cache-hit ratios, such as random I/O activity, and with high write-to-read ratios.

RAID-5 Distributed Parity

RAID-5 disk arrangements consist of four disks (3D+1P) or eight disks (7D+1P). Data is striped across disks similar to RAID-1+. However, RAID-5 keeps parity information on each stripe of data for fault resilience. If a failure occurs, the contents of the failed block can be recreated by reading back other blocks in the stripe and the parity. Parity information is distributed throughout the array to minimize bottlenecks when rebuilding data from a failed disk. The overhead of RAID-5 is equivalent to one disk drive, regardless of the size of the array group. RAID-5 is best suited to applications using mostly sequential reads.

RAID-6 Redundant Distributed Parity

Like RAID-5, RAID-6 stripes blocks of data and parity across an array of drives. However, RAID-6 maintains redundant parity information for each stripe of data. This redundancy enables RAID-6 to recover from the failure of up to two drives in an array: a double fault. Other RAID configurations can only tolerate a single fault. As with RAID-5, performance is adjusted by varying stripe sizes. RAID-6 is good for applications using the largest disks and performing many sequential reads.

RAID Rebuild

If a disk in a RAID group fails, the information on the failed drive is rapidly and automatically rebuilt on a hot spare drive. The Universal Storage Platform VM also allow logical volumes to be expanded online, and for administrators to set rebuild priority, based on the daily processing cycle and RAID configuration installed.

Key Considerations

Choosing the right type of RAID implementation for your SQL Server environment requires a thorough knowledge of both SQL Server and SAN. The following best practices incorporate both Microsoft and Hitachi Data System guidelines for SQL Server environments.

- Place database and log files on physically separate RAID groups.
- Place log files on RAID-1+ rather than RAID-5, depending on the log files' capacity and performance requirements. If high write rates are expected, use RAID-1+ to ensure achieving your performance requirements (see next list item).
- Select the appropriate RAID type for database files based on performance, cost and availability requirements. RAID-1+ provides the best performance and availability, but at a higher cost. RAID-5 can provide good performance with slightly lower availability and at a lower cost. RAID-6 can also be considered if higher availability is important enough to justify reduced write performance due to the additional RAID-6 write overhead.
- For cost-sensitive environments, use RAID-5 for transaction logs rather than RAID-1+. In some cases, RAID-5 performs as well as RAID-1+ on the Universal Storage Platform VM for sequential workloads like Microsoft SQL Server log files.
- Create one tempdb file per CPU core and make all files equal in size.
- Place tempdb files on a separate RAID-1+ group from the database and log files with the fastest hard disk drives available. However, with caution, tempdb files can reside on the same RAID group as the database files. To simplify monitoring, always place tempdb files in a dedicated LU, regardless of RAID group placement. tempdb LU usage and growth must also be evaluated prior to deployment and monitored regularly. In addition, for optimal performance during index rebuilds, spread the tempdb files across multiple LUs.
- Due to the high write burst activity of some SQL functions, do not enable the Cache Destage option of the Universal Storage Platform V and Universal Storage VM systems, as the amount of cache available per LU is reduced when this option is enabled.

Drive Considerations

Selecting a drive type is as important as selecting a RAID type for a storage implementation. With Hitachi Universal Storage Platform VM users can choose between Fibre Channel and SATA drives to meet their applications' performance or capacity requirements. Hitachi recommends Fibre Channel drives for solutions where high performance and availability are critical for business operations. SATA drives continue to be the low-cost option for environments where high performance is not critical to business operations and higher capacities are required. Due to performance and availability requirements, Hitachi recommends the use of Fibre Channel drives for SQL Server database, log and tempdb files and SATA drives for secondary copies, dump and scratch space for cost savings.

System Design Considerations for Hitachi Dynamic Provisioning Software

Hitachi Dynamic Provisioning software uses two primary components: a pool of physical storage capacity (HDP pool) and virtual volumes called dynamic provisioning volumes (DP-VOLs).

The HDP pool is composed of normal logical devices (LDEVs) known as pool volumes (Pool-VOL). For performance reasons, the HDP pool normally has multiple pool volumes from multiple array groups. The total physical capacity of a pool is the sum of the capacities of the pool volumes in that particular pool.

After a DP-VOL is associated with a HDP pool, a path definition is created to assign the DP-VOL to an application host server. As the application writes data to the DP-VOL, the Universal Storage Platform V or Universal Storage Platform VM allocates physical capacity from the HDP pool while monitoring multiple thresholds to alert administrators when more physical pool capacity is needed.

A DP-VOL is a dynamic provisioning virtual volume. The term V-VOL is sometimes used for virtual volume; this can be misleading because other sorts of virtual volumes exist, such as the V-VOL used in Hitachi Copy-on-Write Snapshot software. All DP-VOLs are V-VOLs but not all V-VOLs are DP-VOLs.

You can use several methods to create a new DP-VOL or create or modify a pool, including Hitachi Storage Navigator program, Hitachi Device Manager software and Hitachi Command Control Interface (CCI).

Table 3 describes the key tasks performed by a storage administrator using static or dynamic provisioning.

Table 3. Key Storage Provisioning Tasks

<i>Task</i>	<i>Static Provisioning Action</i>	<i>Dynamic Provisioning Action</i>
Select and allocate array groups and LDEVs	Group a number of physical spindles together and select a RAID level appropriate to the application.	Group a number of physical spindles together and select a RAID level appropriate to the applications. These RAID groups are then assembled into one of the dynamic provisioning pools.
Select allocation size	Choose allocation unit sizes appropriate for the application. Typically this would be a number of gigabytes based on the site's selected standard volume sizes. Inefficiencies are inherent in using standard size volumes. These volumes tend to be oversized relative to the user's requirements and therefore fundamentally are underutilized. This step can be a challenging one and is a key aspect of application capacity planning. Few applications continue to operate if their storage capacity runs out, so the only safe way the system administrator can operate is to make the best plan possible and then include an overhead buffer. Overhead figures of 20 percent or more are common recommendations in this planning stage. Analysts report measured underutilization numbers in the 50 percent or more range.	Hitachi Dynamic Provisioning software has physical allocation unit size that doesn't need to be tuned for the application. The unit of allocation is a 42MB page. Pages are not allocated directly to any volume; they are allocated on use. This is totally transparent to the user. Any volume size can be easily created without detracting from manageability and therefore can improve utilization. Instead of pre-allocating a fixed amount of space, the administrator specifies the maximum size that virtual volume can ever reach. This is a value up to 4TB.
Mapping volumes	These standard size units are mapped to servers and used in applications. At this point all disk space equal to the volume size is consumed.	The virtual volumes are mapped to servers and used in applications. At this point no disk space is consumed.

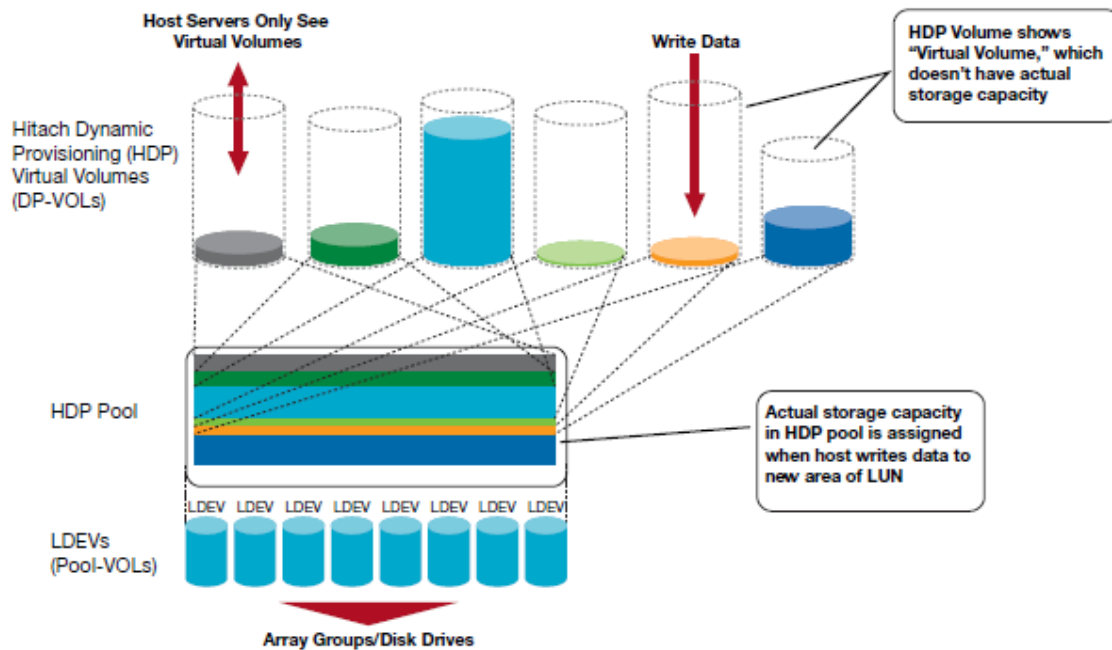


<i>Task</i>	<i>Static Provisioning Action</i>	<i>Dynamic Provisioning Action</i>
Formatting volumes	The storage is formatted and accessed in the usual way.	The storage is formatted and accessed in the usual way, but space is only allocated from the pool when the server writes to the disk.
Capacity utilization and growth	<p>As storage needs grow, administrators repeat parts of this process to add more storage for the application.</p> <p>Unfortunately, another LU is rarely what applications want. All too often DBAs think, “if only I had chosen a larger LU in the first place.”</p> <p>Some methods are available for easing this, specifically online Logical Unit Size Expansion (LUSE).</p>	<p>As storage needs grow, administrators would not need to allocate additional storage; it would be allocated on use from the pool.</p> <p>As the pool is consumed, additional storage capacity can be procured and added nondisruptively.</p>
Data relocation	<p>Storage administrators at this point can be presented with a challenging landscape — they have the amount of storage they require, but the data is often in the wrong location.</p> <p>A long, involved process of data relocation might be required to get everything correct.</p>	<p>Data movement is rarely needed.</p> <p>Data relocation might still be required, but it is more likely to be based on organizational requirements.</p>
Performance leveraging	<p>Even where storage capacity is adequate, performance balance is rarely correct.</p> <p>Even if initially designed correctly, environment changes soon prevent it remaining so.</p> <p>To maintain the required level of performance at a reasonable cost requires further data relocation.</p> <p>Some methods are available for easing this; specifically, Hitachi Tiered Storage Manager software permits online control over volume placement.</p>	<p>One of the benefits of Hitachi Dynamic Provisioning is performance leveling.</p> <p>Performance hot spots are naturally smoothed out.</p>

With Hitachi Dynamic Provisioning, you only use and always have available exactly the storage you need. But a new storage solution inevitably implies some changes in storage management. This is because the design of operating systems, file systems and applications were not originally optimized for dynamic provisioning

Figure 2 shows how Hitachi Dynamic Provisioning automatically spreads the I/O load of all applications accessing the common pool across the available spindles.

Figure 2. Hitachi Dynamic Provisioning — Virtual Volumes



Configuration Options

The Hitachi Universal Storage Platform VM offers a number of ways to provision storage for use in your SQL server environment. The following sections describe the three most common methods.

Static Provisioning

The traditional method utilized for creating SQL Server 2008 database volumes and files remains the same when utilizing Hitachi's Universal Storage Platform VM. Establish a detailed plan prior to deployment to account for future planned and unscheduled or emergency growth.

When an LDEV is created for this type of implementation, all of its capacity becomes dedicated for the environment for which the LDEV is assigned. On average, this leads to between 30 and 50 percent of unutilized capacity being allocated due to a known or expected growth schedule for the database environment. As the need for unscheduled growth arises, you either have to provision new LDEVs and add secondary database files (.ndf) for the environment or utilize a LUSE method. LUSE enables you to create virtual, expandable LDEVs from physical LDEVs in order to support LDEV expansion.

Single Pool Dynamic Provisioning

In this scenario, database and transaction log files are placed in the same pool. When planned and deployed properly, this approach enables you to account for workloads from multiple SQL Server instances while minimizing the overall requirement for both capacity and spindles for a given instance. It also enables you to account for future growth even with database and transaction log volumes in the same pool. This approach provides a better way to utilize your capacity when compared to the multiple pools approach described in the next section. The difference is even more prominent when compared to static provisioning. Most importantly, this approach provides the capability of utilizing a larger amount of spindles for the environment, which allows for an increased amount of IOPS capability and a possible decrease in the latency when compared to the static provisioning approach. All of these benefits are achieved while still ensuring availability, performance and ease of management for your SQL Server environment.

From a volume deployment perspective, testing in a lab environment has shown that a static provisioning deployment from the ground up can take longer than a dynamic provisioning deployment. Similar time savings can also be observed when increasing a volume's capacity from its original settings (for example, increasing a DP-VOL size versus increasing a static LDEV size using a method such as LUSE). While tests have shown an

overall increase in performance in comparison to the static provisioning approach, it is important to note that, as with any SQL deployment plan, a thorough study of the I/O requirements for all databases that will use the dynamic provisioning pool is required to establish the correct minimum number of spindles to be used for the dynamic provisioning environment.

This approach requires additional attention when initially creating the database files. For the database files, the values of SI ZE, FI LEGROWTH and MAXSI ZE are determined by the capacity plan. A trade-off exists between space consumed and performance. Because the dynamic provisioning environment works with a 42MB allocation page scheme, FI LEGROWTH values should be a multiple of 42MB. Additionally, give careful consideration to the backup model selected for a given database and its relationship to how the transaction log files operate around truncation. For example, selecting the simple recovery model ensures that log space is reclaimed automatically to keep requirements small, therefore minimizing the need to manage the transaction log space. For more information about recovery models, see the [Recovery Models and Transaction Log Management](#) page of SQL Server 2008 Books Online.

Multiple Pool Dynamic Provisioning

In this scenario, database and transaction log files are placed in separate pools. In environments where multiple databases' highest transaction levels occur at the same time, this approach can meet the requirements for your environment. When planned and deployed properly, this approach can enable you to implement your environment in a more traditionally recommended manner by separating database and transaction log files into their own dedicated dynamic provisioning pools. While this approach can also account for performance peaks from multiple SQL Server instances while minimizing the overall requirement for both capacity and spindles for a given instance, it is important to note that it should also include a cache partitioning implementation to separate the pool workloads. This approach also enables you to account for future growth. This is accomplished with the use of different pools for database and transaction log volumes that have opposite I/O workload types. This approach also enables you to increase the granularity at which you monitor the performance of both database and transaction log volumes given that they each have their dedicated pool and array groups. While this approach provides smaller savings in capacity utilization, it still provides a better way to utilize capacity when compared to static provisioning while still ensuring availability, performance and ease of management.

Like the single pool approach, this approach requires additional attention when initially creating the database and its files. For the database files, the values of SI ZE, FI LEGROWTH and MAXSI ZE are determined by the capacity plan. A trade-off exists between space consumed and performance. Because the dynamic provisioning environment works with a 42MB allocation page scheme, FI LEGROWTH values should be a multiple of 42MB. Additionally give careful consideration to the backup model selected for a given database and its relation to how the transaction log files operate around truncation. For example, selecting the simple recovery model ensures that log space is reclaimed automatically to keep requirements small, therefore minimizing the need to manage the transaction log space. For more information about recovery models, see the [Recovery Models and Transaction Log Management](#) page of SQL Server 2008 Books Online.

Configuring SQL Server in Hitachi Dynamic Provisioning Environments

The SI ZE, FI LEGROWTH and MAXSI ZE configuration parameters set the allocation strategy for SQL Server database and transaction log files. The parameters are illustrated in the following SQL code to create a database when using Hitachi Dynamic Provisioning software:

```
CREATE DATABASE mydb ON PRIMARY (
NAME = mydb1,
FILENAME = 'S:\mountpoint\data1\mydb1.mdf',
SIZE = 420MB,
MAXSIZE = 42000,
FILEGROWTH = 42)
LOG ON (
NAME = mydblog1,
FILENAME = 'S:\mountpoint\log1\mydblog1.ldf',
SIZE = 420MB,
MAXSIZE = 10000,
```

FILEGROWTH = 42)

Each file is created with an initial SI ZE and autoextends by FI LEGROWTH when the currently allocated space runs out, but stops if it reaches MAXSI ZE. This is a perfect structure for use with Dynamic Provisioning software. These parameters have equivalent values in the capacity plan.

For the database, the values of SI ZE, FI LEGROWTH and MAXSI ZE are determined by the capacity plan. A trade-off exists between space consumed and performance. It makes sense for FI LEGROWTH to be a multiple of a page, 42MB, as this improves on-disk contiguity. If SI ZE and FI LEGROWTH are large, then pages are allocated more rapidly, but they have high contiguity. However, contiguity is generally less important with dynamic provisioning than static provisioning. MAXSI ZE is more straightforward, and it is used to make sure that your application does not exceed capacity, fail or both. This is an additional control over your space. Set SI ZE to a value that creates an initial allocation of data that is at least equal to the amount of data you plan to import into the database or that you expect the database or log files to be initially. Pay special attention to the SI ZE and FI LEGROWTH values given to log files, as small sizes might affect system performance. If the log files grow to a large size due to many small increments, they can have lots of virtual log files. This can slow database startup as well as log backup and restore operations for a given SQL Server instance. Microsoft recommends that you assign log files a SI ZE value close to the final size required, and also have a relatively large FI LEGROWTH value (use a multiple of 42MB on dynamic provisioning implementations).

Use the instant file initialization feature of SQL Server to allow faster and optimized data file creation and growth and fast execution of database or filegroup restore operations. Instant file initialization reclaims used disk space without filling that space with zeros. This means that it only overwrites content as new data is written to the files, which makes it an ideal SQL Server feature to work in conjunction with Hitachi Dynamic Provisioning. For more information, see the [Data File Initialization](#) page of SQL Server 2008 Books Online.

Key Considerations

- Always utilize the quick formatting option when formatting Hitachi Dynamic Provisioning volumes, because it is a thin-friendly operation. Slow formatting is equally efficient on Windows 2003 but tests on Windows 2008 show that slow format wrote more data. Slow formatting with RAID systems offers no benefit because all devices are pre-formatted.
- Use SI ZE, FI LEGROWTH and MAXSI ZE configuration parameters that set the allocation strategy for its database and transaction log files for use with Hitachi Dynamic Provisioning. Use values that are equal to or multiples of 42MB.
- Use the instant file initialization feature of SQL Server as it is both a thin friendly operation as well a best practice approach for faster data file creation and growth.
- Due to its workload nature, the log file receives burst-sequential write traffic with the same disk areas being re-used, so contiguity is beneficial to reduce seek overhead. Aim for moderately large SI ZE and FI LEGROWTH values, such as 420MB.
- Do not defragment file systems on Hitachi Dynamic Provisioning volumes, including those containing database or transaction log files. In a Hitachi Dynamic Provisioning environment, defragmentation of NTFS is rarely space efficient with any data.
- Use a single HDP pool for the database and transaction log volumes for most SQL Server deployments. With this approach, SQL Server workloads can take advantage of additional spindles being available due to dynamic provisioning wide striping capability. This provides capacity savings and ease of management while ensuring the highest level of performance required by your environment.
- Use a separate HDP pool for database and transaction log volumes in environments where a more traditional SQL Server deployment is needed. Consider a cache partitioning implementation when implementing this type of environment. This also provides capacity savings and ease of management while ensuring the highest level of performance required by your environment.

- Use at least four array groups for each dynamic provisioning pool. Each array group should have a single LDEV that utilizes the full capacity of the array group; the LDEVs are then assembled into one dynamic provisioning pool.
- Use dedicated volumes for both database and transaction log files.

Performance, Availability, Capacity Utilization and Ease of Management Capabilities

Hitachi Dynamic Provisioning ensures the achievement of performance, availability, capacity utilization and ease of management by enabling DBAs and storage administrators to accomplish the tasks that are important to their SQL Server environment.

Performance Factors

Tests show that the performance design and consequent disk design recommendations for dynamic provisioning are similar to static provisioning. For dynamic provisioning, however, calculate the required I/O for the pool, rather than the array group. In addition, the pool performance requirement (number of IOPS) is the aggregate of all applications using the same pool. The same is true if a static array group supports multiple applications.

The volume performance feature is an automatic result from the manner in which the individual HDP pools are created. A pool is created using 1-1024 LDEVs (pool volumes) that provide the physical space, and the pool's 42MB allocation pages are assigned on demand to any of the DP-VOLs connected to that pool. Each individual 42MB pool page is consecutively laid down on a whole number of RAID stripes from one pool volume. Other pages assigned over time to that DP-VOL will randomly originate from the next free page from other pool volumes in that pool.

As an example, assume that an HDP pool is assigned 24 LDEVs from 12 RAID-1+ (2D+2D) array groups. All 48 disks contribute their IOPS and throughput power to all of the DP-VOL assigned to that pool. If more random read IOPS horsepower is desired for that pool, it can be created with 64 LDEVs from 32 RAID-5 (3D+1P) array groups, thus providing 128 disks of IOPS power to that pool. You can also increase the capacity of an HDP pool by adding array groups to it, thus re-leveling the wide striping along the pool and contributing its IOPS and throughput power to all of the DP-VOL assigned to that HDP pool.


Up to 1024 such LDEVs can be assigned to a single pool. This can represent a considerable amount of I/O power under (possibly) just a few DP-VOLs. This type of aggregation of disks was only possible previously by the use somewhat complex host-based volume managers (such as Veritas VxVM) on the servers. One alternative available on both the Universal Storage Platform V and Universal Storage Platform VM is to use the LUSE feature, which provides a simple concatenation of LDEVs. Unlike Hitachi Dynamic Provisioning, however, the LUSE feature is mostly geared towards solving only capacity problems rather than both capacity and performance capability problems.

High Availability

In addition to offering the most scalable and highest performing platform for enterprise storage, the Universal Storage Platform V and Universal Storage Platform VM place maximum emphasis on high-availability computing to satisfy the needs of the most demanding enterprise applications. This enables any solution implemented on the Universal Storage Platform V and Universal Storage Platform VM to be highly available, whether you use choose static or dynamic provisioning for your SQL Server 2008 environment.

Capacity Utilization and Unscheduled Capacity Growth Capability

Hitachi Dynamic Provisioning enables you to proficiently utilize the capacity available in your Universal Storage Platform VM system while provisioning volumes for future needs and ensuring overall performance for your SQL Server 2008 environment. Through the assignment of multiple array groups to a given pool, the Universal Storage Platform VM ensures that your SQL Server 2008 environment performs according to your application needs while utilizing the space currently required from each of the created volumes and accounting for scheduled growth.



While a well-planned deployment of SQL Server environments utilizing Hitachi Dynamic Provisioning can help eliminate the need for purchasing capacity that is not utilized upon deployment, you cannot take into consideration unexpected growth factors that are beyond your forecast ability. If such situations arise, you can still increase a given Hitachi Dynamic Provisioning volume size through the use of Hitachi Command Control Interface (CCI) `rai dvchkset` command. The operation can be achieved for as long as free space is available within the HDP pool of the volume for which you are trying to increase capacity, the DP-VOL being expanded is not in a paired state with Hitachi ShadowImage® Replication software, Hitachi TrueCopy® Synchronous software or Hitachi Universal Replicator software, and the SQL Server host's operating system is Windows Server 2008. Give careful consideration to volumes that are part of a replication or mirroring environment as their secondary volumes also need increased capacity to continue the business continuity operation.

Ease of Management

The Universal Storage Platform V and Universal Storage Platform VM implementation of Hitachi Dynamic Provisioning is unique in that all internal and externally attached storage resources virtualized by the Hitachi storage controller can participate in the common physical pool used by DP-VOLs. The storage system manages wide striping dynamically, reducing the need for storage administrators to manually tune I/O for performance. One of the most significant benefits that you might experience from deploying Dynamic Provisioning software is in managing storage allocations. Additional benefits might also include the following:

- Discontinuing data micro-management of data volumes and data re-allocation and re-provisioning due inadequate available space
- No longer needing to affect application performance due to the allocation of too many volumes on the same drive group
- Discontinuing unnecessary allocation of space that is likely to not be utilized for months or years

Host Considerations

To ensure optimal performance from your storage system, it is important to understand disk alignment, NTFS allocation unit size, path redundancy, HBA placement and queue depth during the planning process.

Disk Alignment

Both disk alignment offset and NTFS allocation unit size must be set when LUs are partitioned and formatted at the operating system level before Microsoft SQL Server database files are created. Windows Server 2008 eliminates the need to align the partition, due to the use of 1024K as the default partition offset. For Windows Server 2003, use Diskpart.exe to ensure that the disk tracks are sector and cache aligned. Use an offset of 64KB (128 sectors) for every disk that is part of the SQL environment. It is important to remember that this is a destructive process and must be performed before database files are created to prevent data loss.

NTFS Allocation Unit Size


Choose an allocation unit size that matches the I/O profile of your application to allow more efficient I/O on your disk subsystem. When formatting your SQL LUs in Windows Server, override the default setting and specify an allocation unit size of 64KB. To guarantee that the minimum space is written on a virtual pool, always use quick formatting when using Hitachi Dynamic Provisioning.

Host Bus Adapter Drivers

When choosing HBA drivers, ensure that you are using the current recommended drivers for the Universal Storage Platform VM. Major HBA vendors allow you to download current drivers for Hitachi storage systems. For a list of currently supported HBAs and drivers, see the Interoperability Information section on the Hitachi Data Systems Web site.

HBA Queue Depth Settings

Queue depth settings determine how many command data blocks can be sent to a port at one time. Setting queue depth too low can artificially restrict an application's performance, while setting it too high might cause a slight reduction in I/O. Setting queue depth correctly allows the controllers on the Hitachi storage system to optimize multiple I/Os to the physical disk. This can provide significant I/O improvement and reduce response time.



Applications that use SQL Server are I/O intensive and can have many concurrent, outstanding I/O requests. For that reason, better performance is generally achieved with higher queue depth settings. However, this must be balanced with the available command data blocks on each front-end port of the storage system.

The Universal Storage Platform VM has a maximum of 2048 command data blocks available on each front-end port. This means that at any one time up to 2048 active host channel I/O commands can be queued for service on a front-end port. The 2048 command data blocks on each front-end port are used by all LUs presented on the port, regardless of the connecting server. When calculating queue depth settings for your Microsoft SQL Server HBAs, you must also consider queue depth requirements for other LUs presented on the same front-end ports to all servers. Hitachi recommends setting HBA queue depth on a per-target basis rather than per-port basis.

To calculate queue depth, use the following formula:

$$2048 \div \text{total number of LUs presented through the front-end port} = \text{HBA queue depth per host}$$

For example, suppose that four servers share a front-end port on the storage system, and between the four servers, 16 LUs are assigned through the shared front-end port and all LUs are constantly active. The maximum dynamic queue depth per HBA port is 128, that is:

$$2048 \text{ command data blocks} \div 16 \text{ LUs presented through the front-end port} = 128 \text{ HBA queue depth setting}$$

SAN Data Path Redundancy

Hitachi recommends the use of dual SAN fabrics, multiple HBAs and host-based multipathing software when deploying business critical SQL Server applications. Two or more paths from the SQL Server connecting to two independent SAN fabrics is essential for ensuring the redundancy required for critical applications. The Universal Storage Platform VM supports up to 48 Fibre Channel ports from up to three front-end director modules that support both direct connect as well as multiple paths with the use of a Fibre Channel switch. Unique port virtualization technology dramatically expands connectivity from Windows Server to the Universal Storage Platform VM. Each physical Fibre Channel port supports 1024 virtual ports.

Multipathing software such as Hitachi Dynamic Link Manager and Microsoft Windows Server 2008 native MPIO are critical components of a highly available system. Multipathing software allows the Windows operating system to see and access multiple paths to the same LU, enabling data to travel down any available path for increased performance or continued access to data in the case of a failed path. Hitachi Dynamic Link Manager includes the following load balancing algorithms that are especially suited for Hitachi storage systems:

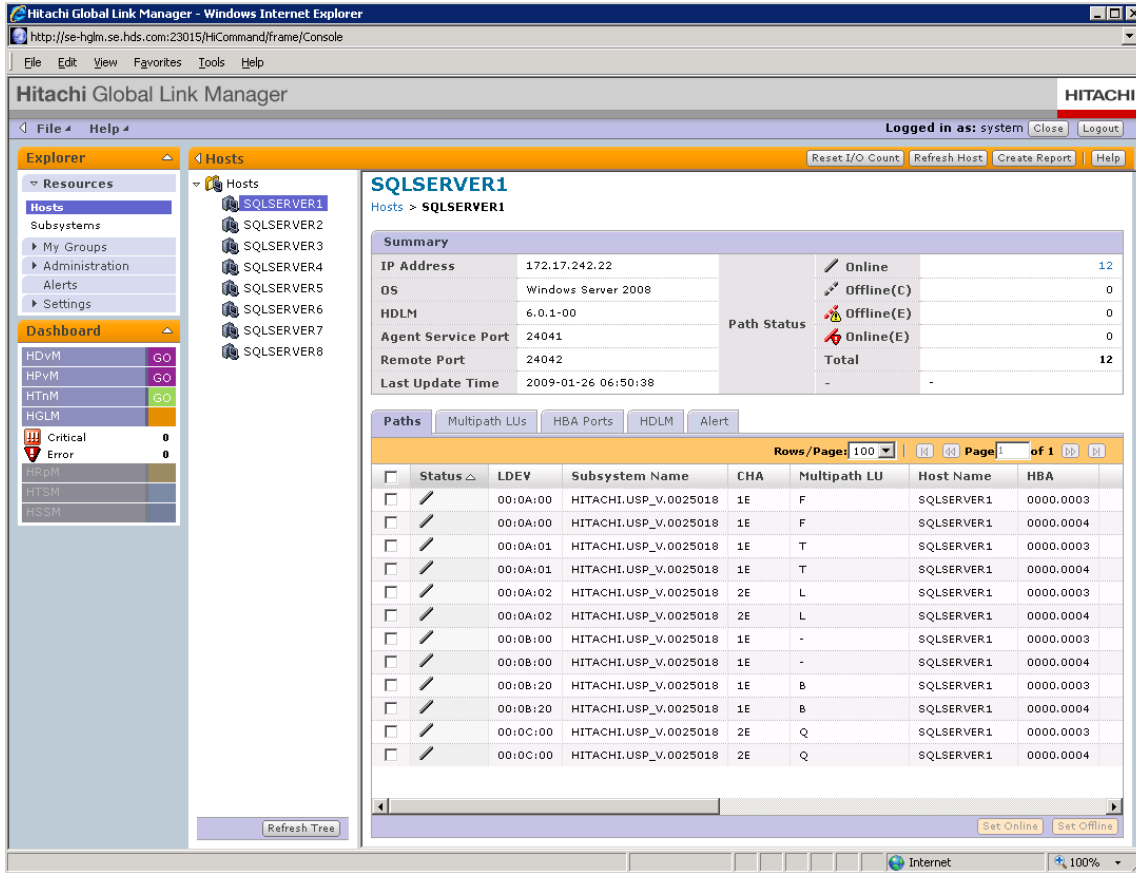
- Round robin
- Extended round robin
- Least I/Os
- Extended least I/Os
- Least blocks
- Extended least blocks

While previous tests showed extended round robin to be the best algorithm to use in SQL Server environments, the introduction of additional load balancing algorithms increase the choices that can increase the performance of your SQL Server 2008 environment. In test environments, both extended least I/Os and extended least blocks are good candidates for SQL Server environments. Like any SQL Server deployment, conduct testing before establishing which of the two algorithms is better suited for your SQL Server 2008 environment.

Hitachi Global Link Manager software consolidates, simplifies and enhances the management, configuration and reporting of multipath connections between servers and storage systems. Hitachi Global Link Manager software manages all of the Hitachi Dynamic Link Manager installations in the environment. Use it to configure multipathing on the SQL Server hosts, monitor all the connections to the USP VM storage system, and to

report on those connections. Global Link Manager also enables administrators to configure load-balancing on a per-LU level. Hitachi Global Link Manager software also integrates with the Hitachi Storage Command Suite of products and is usually installed on the same server as Hitachi Device Manager. Figure 3 shows the Hitachi Global Link Manager interface.

Figure 3. Hitachi Global Link Manager Interface



Key Considerations

- Use a 64KB allocation unit size for database, transaction logs, tempdb and filestream volumes.
- Use the most current Hitachi supported HBA drivers.
- Select the proper HBA queue depth using the formula described above.
- Test both extended least I/Os and extended least blocks algorithms to establish which is better suited for your SQL Server environment. Alternatively, you can use round robin as the load-balancing setting on Windows Server 2008 native MPIO.
- Use at least two HBAs and place them on different buses within the server to distribute the workload over the server's PCI bus architecture.
- Use at least two Fibre Channel switch fabrics to provide multiple independent paths to the Universal Storage Platform VM to prevent configuration errors from bringing down the entire SAN infrastructure.
- Logs, data and tempdb can share the same front end ports on the storage system, however, keep these ports exclusive to SQL Server if possible. If the ports must be shared with other servers, ensure that the servers not dedicated to SQL Server have low I/O requirements for the shared ports.

Performance Monitoring

A complete, end-to-end picture of your SQL Server environment and continual monitoring of capacity and performance are key components of a sound database management strategy. Monitor servers, operating systems, SQL Server instances, databases, database applications, storage and IP networks, and the Universal Storage Platform VM using tools such as Windows Performance Monitor (PerfMon), Hitachi Performance Manager feature, Hitachi Tuning Manager and SQL Server, Dynamic Management Views (DMV) and Dynamic Management Functions (DMF).

Note that while PerfMon provides good overall I/O information about a SQL Server environment and specific SQL Server performance characteristics, it cannot identify all possible bottlenecks in an environment. For a good overall understanding of the I/O profile of a given SQL Server host, monitor the storage system's performance with Hitachi Performance Manager feature. Combining data from at least two performance-monitoring tools provides a more complete picture of the SQL Server environment. Remember that PerfMon is a per-server monitoring tool and cannot provide a holistic view of the storage system. For a complete view, use PerfMon to monitor all servers that are sharing a RAID group.

Windows Performance Monitor

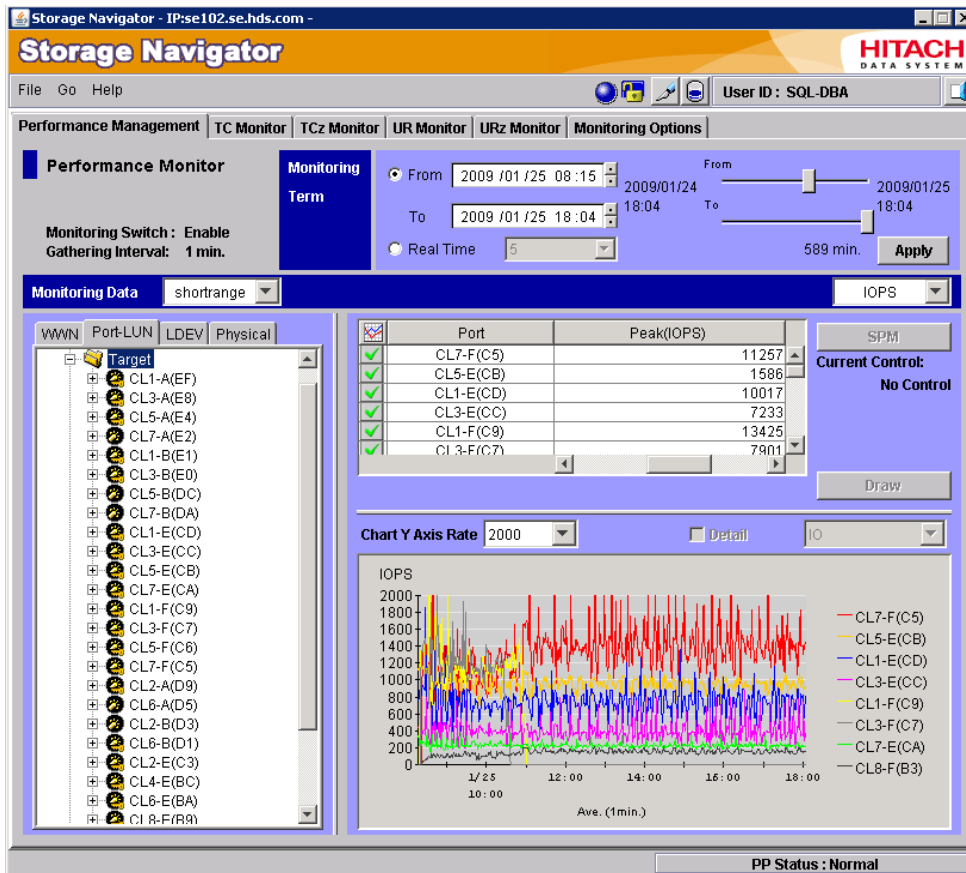
PerfMon is a Windows-based application that allows administrators to monitor the performance of a system using counters or graphs, in logs or as alerts on the local or remote host. SQL Server performance counters are included in PerfMon's counters. Use these counters to identify and troubleshoot possible bottlenecks that the SQL Server might be experiencing. For more information about monitoring host-related counters, see the "Monitoring I/O Performance Using System Monitor" section of Microsoft's [Predeployment I/O Best Practices for Microsoft SQL Server](#) article on TechNet.

Hitachi Performance Manager Feature

Hitachi Performance Manager feature is a controller-based software application, enabled through Hitachi Enterprise Storage Navigator, which monitors the performance of RAID groups, logical units and other elements of the disk subsystem while tracking utilization rates of resources such as hard disk drives and processors (see Figure 3). Information is displayed using line graphs in the Performance Manager windows and can be saved in comma-separated value (.csv) files.

When the disk subsystem is monitored using Hitachi Performance Manager feature, utilization rates of resources in the disk subsystem (such as load on the disks and ports) can be measured. When a problem such as slow response occurs in a host, an administrator can use Hitachi Performance Manager feature to quickly determine if the disk subsystem is the source of the problem. Figure 4 shows the Hitachi Performance Manager Feature interface.

Figure 4. Hitachi Performance Manager Feature



Hitachi Tuning Manager

Hitachi Tuning Manager software provides intelligent and proactive end-to-end monitoring, reporting and forecasting capabilities of storage resources with a focus on business applications such as Microsoft, SQL Server, and Microsoft Exchange. Fully integrated with the Hitachi Storage Command Suite, Basic Operating System, and Device Manager software, and compliant with industry standards, the Tuning Manager module can be used with Hitachi Universal Storage Platform V and Universal Storage Platform VM storage systems.

Gaining a complete view of the I/O path, including software, servers, HBAs, switches, switch and storage subsystem ports, storage subsystem and disks that collectively make up the SQL Server infrastructure is essential to understanding and managing performance. But accessing this information is no easy task. Few tools offer a sufficiently detailed picture for administrators to state with certainty the cause of poor response times.

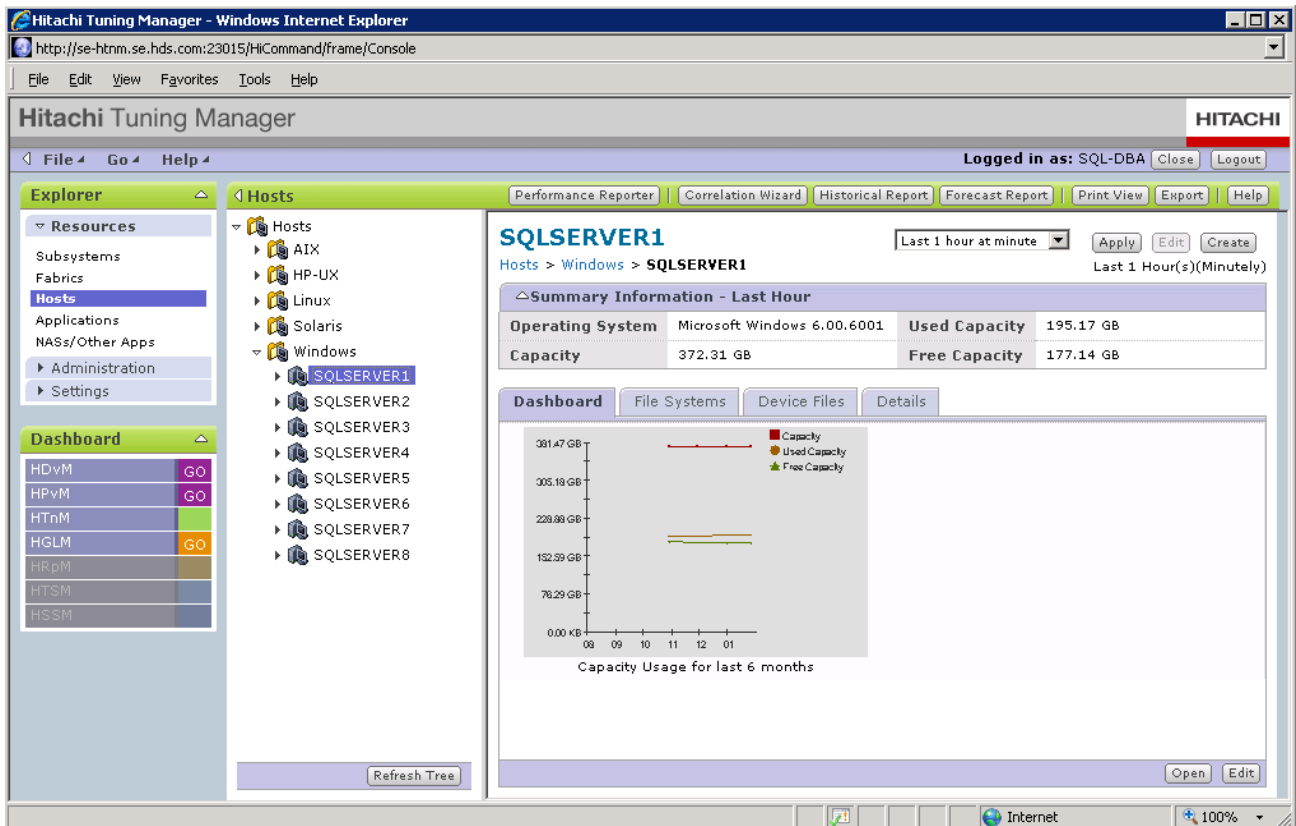
Hitachi Tuning Manager software offers full application-to-spindle performance monitoring, including an agent specifically for monitoring SQL Server databases. While the agent for SQL Server 2005 is currently available, the agent for SQL Server 2008 will be available with Hitachi Tuning Manager 6.2. It tracks more than 150 performance metrics within SQL Server, Microsoft Window Server hosts, SAN switches and within Hitachi storage systems. Making product effective in managing SQL Server databases, however, requires some best practices to know what to look for. Hitachi Data Systems Global Solution Services (GSS) offers a custom reporting service based on Tuning Manager, applying the best practices that allow customers to monitor and manage their SQL Server applications. Combined, these provide administrators the tools and knowledge to ensure storage systems used by the enterprise for database applications remain fully optimized.

Hitachi Tuning Manager software provides insight into the enterprise IT environment along the entire path taken by application I/O. An automated, intelligent, and path-aware storage resource management tool, Tuning

Manager software maps, monitors, analyzes, and reviews storage network resources in real time, giving administrators instant access to the current state of hosts, file systems, SQL Server databases, SANs and storage systems.

Using agent software installed on platforms in the path of application I/O, Tuning Manager software is able to provide a comprehensive assessment of the performance and capacity of the entire I/O infrastructure. Individual alarms can be set for data volumes that are more critical, or most likely to grow rapidly, alerting you to sudden changes or possible risks of shortfall in a given dynamic provisioning pool. In addition to real-time reports, Tuning Manager software compiles metrics in a historical database allowing administrators to compare current and past behavior to quickly identify anomalies. Figure 5 shows the Hitachi Tuning Manager interface.

Figure 5. Hitachi Tuning Manager



SQL Server Data Management Views and Functions

SQL Server uses DMV and DMF information to provide DBAs with instrumentation to monitor their SQL Server environments. DMV and DMF information contains useful views and functions that enable users not only to monitor the health of their SQL Server instance, but most importantly provide useful information that enables DBAs to diagnose and tune performance related problems on the database design. Using the DMV information, DBAs can obtain extremely granular information on databases, query statistics, I/O statistics and operating system level information such as stacks, tasks and threads currently running on the SQL server.

While SQL Server 2008 specific Microsoft TechNet articles around performance problems as well as waits and queues were unavailable at the time this paper was published, some generic information pertaining to troubleshooting SQL Server can be found at [Troubleshooting Performance Problems in SQL Server 2005](#) and [SQL Server 2005 Waits and Queues](#).

Table 4 provides an overall view of the DMV and DMF information available on SQL Server. For more information, including a complete list of all available DMVs and DMFs, see the [Dynamic Management Views and Functions \(Transact-SQL\)](#) document available at Microsoft TechNet's Web site.

Table 4. Prefixes for DMVs and DMFs

<i>Prefix</i>	<i>Description</i>
dm_clr*	Common language runtime-related DMV
dm_db*	General database statistics
dm_exec*	Execution related DMV
dm_fts*	Full-text search related DMV
dm_io*	I/O related DMV
dm_qn*	Query notification related DMV
dm_repl	Replication related DMV
dm_broker*	Service broker related DMV
dm_os*	SQL Server operating system related DMV
dm_tran*	Transaction related DMV

With SQL Server 2008, Microsoft introduces five new DMVs to present memory information. For more information about each of these new DMVs and their respective columns, see the following information on Microsoft's Web site:

- [sys.dm_os_memory_brokers \(Transact-SQL\)](#)
- [sys.dm_os_memory_nodes \(Transact-SQL\)](#)
- [sys.dm_os_nodes \(Transact-SQL\)](#)
- [sys.dm_os_process_memory \(Transact-SQL\)](#)
- [sys.dm_os_sys_memory \(Transact-SQL\)](#)

It is important to note that dynamic management views and functions return internal, implementation-specific state data. This means that the schema and data they return might change in future releases of SQL Server at Microsoft's discretion. For that reason, keep in mind that future releases of DMV and DMF might not be compatible with their current SQL Server 2008 release.

SQL Server 2008 Management Data Warehouse

Management Data Warehouse (MDW) is a new SQL Server 2008 feature. MDW is a relational database that contains the data collected from a SQL server that is a data collection target. This data is used to generate the reports for the system data collection sets and can also be used to create custom reports. A good understanding of how SQL Server 2008 collects performance and diagnostic data and presents the information in Management Studio is necessary for a successful deployment of MDW.

You can install MDW on the same instance of SQL Server that runs the data collector. However, if server resources or performance are issues on the server being monitored, you can install MDW on a different computer. For more information, see the [Getting Started with the Data Collector](#) page of SQL Server 2008 Books Online.



References

[Hitachi Universal Storage Platform™ V and Hitachi Universal Storage Platform™ VM Architecture Guide](#) — Using a Services Oriented Storage approach to anticipate the need to adapt quickly to change and accommodate massive capacity growth

[Hitachi Dynamic Provisioning Software Best Practices Guide](#) — Guidelines for the Use of Hitachi Dynamic Provisioning Software with the Microsoft Windows Operating System and Microsoft SQL Server 2005, Microsoft Exchange 2003 and Microsoft Exchange 2007 by Steve Burr

[Tuning Microsoft SQL Server 2005 Performance](#) — Using Hitachi Tuning Manager to monitor and tune SQL Server environments by Joe Carlisle

Appendix A: Using Performance Monitoring Counters

Best practice is to establish a baseline for your SQL Server environment using performance monitoring tools and then monitor and analyze the environment on an ongoing basis.

Table 5 describes the key PerfMon counters to use when monitoring and analyzing disk-related interactions between SQL Server hosts and the storage system.

Table 5. Key PerfMon Counters for Host Storage System Analysis

<i>Performance Monitor Counter</i>	<i>Microsoft Description</i>	<i>Hitachi Data Systems Notes</i>
Disk Reads/sec Disk Writes/sec	Number of I/Os per second (IOPs) being issued against a particular disk or volume. This number varies based on the size of I/Os issued. Consult the hardware vendor for an estimation of the number of I/Os per second support per spindle on their particular hardware.	Total number of read or write operations taking place per second for the selected logical or physical disk. This field indicates the quantity of I/O, not the size of the I/O.
Average Disk/sec Read Average Disk/sec Write	Measure of disk latency. Lower values are better but this can vary and is dependent on the size and nature of the I/Os being issued. Numbers also vary across different storage configurations (cache size/utilization can impact this greatly). On well-tuned I/O subsystems, ideal values would be: 1–5 ms for Log (ideally 1 ms on arrays with cache) 4–20 ms for Data on OLTP systems (ideally 10 ms or less) 30 ms or less on DSS (decision support system) type. Latencies here can vary significantly depending on the number of simultaneous queries being issued against the system. Sustained values of more than this when the total throughput is less than expected should be investigated. Consider these in combination with what is normal for your particular system. Make sure to monitor disk latencies for trend analysis. The number of I/Os and latency specific to SQL Server data files can be found by using the <code>sys.dm_io_virtual_file_stats</code> dynamic management view in SQL Server 2005.	Average response time in milliseconds for the selected logical or physical disk. This average can be affected by I/O size, RAID configuration and other factors in the data path.
Average Disk Bytes/Read Average Disk Bytes/Write	Size of I/Os being issued. Impacts disk latency. Large I/O sizes may result in slightly higher latency. This will tell you the average size of the I/Os SQL is issuing to fill query requests.	Average size in bytes of the I/O being issued.
Average Disk Queue Length	Average number of outstanding I/O requests. The general rule of thumb is ≤ 2 per spindle but this may be hard to measure due to storage virtualization, differences in RAID level between configurations, and so on. Focus on higher than average disk queue length in combination with higher than average disk latencies. This combination could indicate that the storage array cache is being over utilized or spindle sharing with other applications is impacting performance.	A high average disk queue length in combination with a high average disk/sec read (response time) might indicate that the HBA queue depth is set incorrectly for your environment. The controller on the storage system might be able to further optimize physical I/O when using an increased queue depth.

<i>Performance Monitor Counter</i>	<i>Microsoft Description</i>	<i>Hitachi Data Systems Notes</i>
Disk Read Bytes/sec Disk Write Bytes/sec	Measure of the total bandwidth for a particular disk or LU.	Total number of bytes either read or written per second for the selected logical or physical disk.

Table 6 describes the key counters to use when monitoring and analyzing SQL Server-related operations.

Table 6. Key PerfMon Counters for SQL Server-related Operations

<i>Category</i>	<i>Performance Monitor Counter</i>	<i>Microsoft Description</i>	<i>Hitachi Data Systems Notes</i>
Buffer Manager	Buffer cache hit ratio	Percentage of pages found in the buffer cache without having to read from disk. The ratio is the total number of cache hits divided by the total number of cache lookups over the last few thousand page accesses. After a long period of time, the ratio moves very little. Because reading from the cache is much less expensive than reading from disk, you want this ratio to be high. Generally, you can increase the buffer cache hit ratio by increasing the amount of memory available to SQL Server.	A high number indicates that the system is mainly utilizing cached data for its operations rather than going to the physical disk for data.
	Page life expectancy	Number of seconds a page will stay in the buffer pool without references.	As the server requests more memory for other processes, this value decreases.
	Page reads/sec	Number of physical database page reads that are issued per second. This statistic displays the total number of physical page reads across all databases. Because physical I/O is expensive, you may be able to minimize the cost, either by using a larger data cache, intelligent indexes and more efficient queries, or by changing the database design.	This counter is for all databases, including tempdb.
	Page writes/sec	Number of physical database page writes issued per second.	Total number of pages written, including split pages.
	Read ahead pages/sec	Number of pages read per second in anticipation of use.	The SQL Server engine looks at the type of query being performed and makes decisions as to what pages to read ahead of time. This can lead to a deceptively small buffer hit rate.
Databases	Percent log used	Percentage of space in the log that is in use.	High log space utilization rates indicate a need for more storage resources for logs, that logs need to be backed up, or that they are not being truncated.



<i>Category</i>	<i>Performance Monitor Counter</i>	<i>Microsoft Description</i>	<i>Hitachi Data Systems Notes</i>
	Transactions/sec	Number of transactions started for the database per second.	The transactions per second and the amount of pages being read or written indicate the actual workload on SQL Server. OLTP is characterized as more reads than writes. Transactions per second and batch requests per second are an indication of server usage.
General Statistics	Processes blocked	Number of currently blocked processes.	This counter monitors processes waiting for pages that are locked by other processes. The lock might be a page, table or database lock preventing a process from accessing a page. A high number of blocked processes does not necessarily indicate I/O fault.
Latches	Average latch wait time (ms)	Average latch wait time (in milliseconds) for latch requests that had to wait.	This counter indicates the average number of milliseconds the latches waited, regardless of the reasons for the delays.
	Latch waits/sec	Number of latch requests that could not be granted immediately.	When transactions and batch requests are high and latch waits per second are low, the server is operating efficiently.
	Total latch wait time (ms)	Total latch wait time (in milliseconds) for latch requests in the last second.	
Locks	Average wait time (ms)	Average amount of wait time (in milliseconds) for each lock request that resulted in a wait.	High average wait time might be an indication of a page, table or database lock.
	Lock wait time (ms)	Total wait time (in milliseconds) for locks in the last second.	
Memory Manager	Memory grants pending	Total number of processes waiting for a workspace memory grant.	Even if a process has all the pages it needs, it might still be waiting for a workspace to complete a process. A high number can be an indication that many concurrent queries need workspace memory grants at the same time. Some examples of queries supported by workspace grants are sorts, hash joints and aggregations.
SQL Statistics	Batch requests/sec	Number of Transact-SQL command batches received per second. This statistic is affected by all constraints (such as I/O, number of users, cache size, complexity of requests and so on). High amount of batch requests means good throughput.	Batch requests and transactions per second indicate how busy the server is.

<i>Category</i>	<i>Performance Monitor Counter</i>	<i>Microsoft Description</i>	<i>Hitachi Data Systems Notes</i>
	SQL Compilations/sec	Number of SQL compilations per second. Indicates the number of times the compile code path is entered. Includes compiles caused by statement-level recompilations in SQL Server 2005. After SQL Server user activity is stable, this value reaches a steady state.	Compiles and recompiles are caused by changes to the environment. If these counters are high, it is an indication that something is causing the system's environment to change, like un-parameterized SQL statements, memory pressure on the processor's cache and more. Performing a dbcc freeproccache causes the buffers to flush and all processes to recompile and rerun.
	SQL Re-compilations/sec	Number of statement recompiles per second. Counts the number of times statement recompiles are triggered. Generally, you want the recompiles to be low. In SQL Server 2005, recompilations are statement-scoped instead of batch-scoped recompilations as in Microsoft SQL Server 2000. Therefore, direct comparison of values of this counter between SQL Server 2005 and earlier versions is not possible.	
Wait Statistics	Average wait time (ms)/Lock waits	Average time for processes waiting on a lock.	
	Average wait time (ms)/Log write waits	Average time for log buffer to be written.	
	Average wait time (ms)/Network I/O waits	Average time for wait on network I/O.	
	Average wait time (ms)/Page I/O latch waits	Average time for page I/O latches.	
	Average wait time (ms)/Page latch waits	Average time for page latches, not including I/O latches.	

Use the Hitachi Performance Manager feature counters described in Table 7 when monitoring or analyzing a SQL Server environment. Keep in mind that the performance of the storage system is monitored from the controller level down to each of its main sections (port, RAID groups, logical unit, cache, processor, drive, drive operation and back end). When both controllers are being used, select the following counters from both controllers.

Table 7. Hitachi Performance Manager Feature Counters

<i>Performance Manager Counter</i>	<i>Description</i>	<i>Normal Value</i>
Port IO Rate (IOPS)	Total number of commands (reads and writes) per second on the selected ports. Port read/write I/O rates can be individually monitored as described using other port counters.	Varies
Port Read Rate Port Write Rate (IOPS)	Number of read/write commands per second on the selected ports.	Varies
Port Read Hit Port Write Hit (%)	Indication of cache-hitting within the received read/write command on the selected ports. The read/write hit counters can also be monitored at a RAID group or logical unit level.	Read – Varies ¹ Write – 100%
Port Transfer Rate (MB/sec)	As with the port I/O rates, port read/write transfer rates can also be individually monitored for a given port.	Varies
Parity Group Group IO Rate (IOPS)	Total number of commands (reads and writes) per second on the selected RAID group. As with the port I/O rates, parity group read/write I/O rates can also be individually monitored for a given RAID group.	Varies
Parity Group Group Read Rate RAID Group Write Rate (IOPS)	Number of commands (reads or writes) per second on the selected parity groups.	Varies
Cache Write Pending (%)	Indication of cache being used to buffer writes on the selected controller.	1% to 25% ²
Processor Usage (%)	Processor utilization rate on the selected controller.	1% to 50% ³

¹Port Read Hit counter is highly dependent on the type of workload and overall user load the database has.

² Above 25 percent might be an indication of insufficient spindles for one or more RAID groups or that one or more RAID groups were created on an incorrect RAID type (for example, RAID-5 instead of RAID-1+).

³ If one controller reaches a high utilization rate while the other is underutilized, the Hitachi Dynamic Controller Load Balancing Controller architecture automatically balances workload between controllers.

Appendix B: DMV and DMF Examples

Following are some examples of the type of SQL code that can be written to use DMV and DMF and the output they can provide. Keep in mind that this is a very limited view of DMV and DMF capabilities.

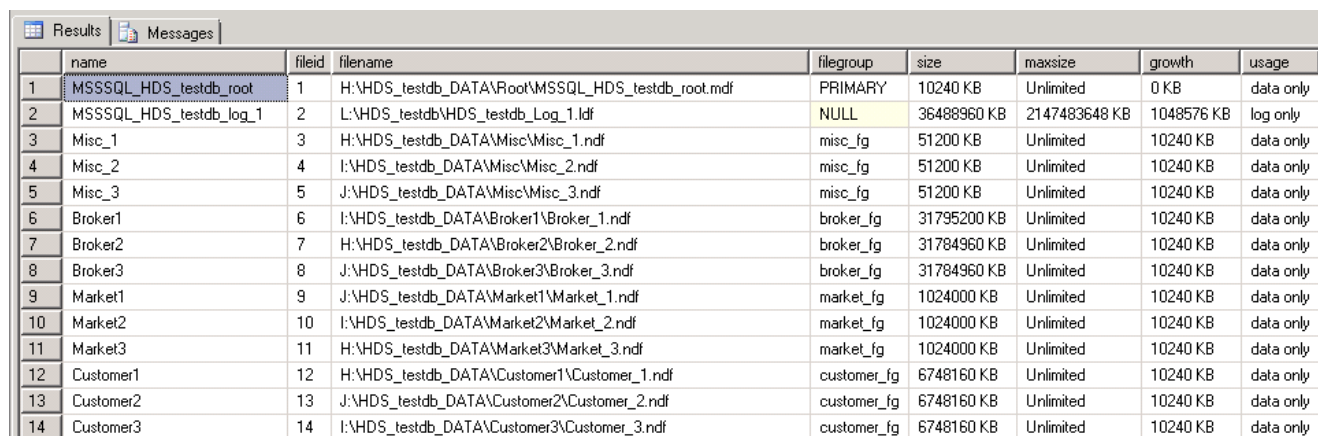
Monitoring Database File Information

To track databases file information, use the `sp_hel pfi l e` stored procedure.

```
use HDS_testdb
go
sp_hel pfi l e
```

Running this SQL code produces the results shown in Figure 6.

Figure 6. `sp_hel pfi l e` Output



	name	fileid	filename	filegroup	size	maxsize	growth	usage
1	MSSQL_HDS_testdb_root	1	H:\HDS_testdb_DATA\Root\MSSQL_HDS_testdb_root.mdf	PRIMARY	10240 KB	Unlimited	0 KB	data only
2	MSSQL_HDS_testdb_log_1	2	L:\HDS_testdb\HDS_testdb_Log_1.ldf	NULL	36488960 KB	2147483648 KB	1048576 KB	log only
3	Misc_1	3	H:\HDS_testdb_DATA\Misc\Misc_1.ndf	misc_fg	51200 KB	Unlimited	10240 KB	data only
4	Misc_2	4	I:\HDS_testdb_DATA\Misc\Misc_2.ndf	misc_fg	51200 KB	Unlimited	10240 KB	data only
5	Misc_3	5	J:\HDS_testdb_DATA\Misc\Misc_3.ndf	misc_fg	51200 KB	Unlimited	10240 KB	data only
6	Broker1	6	I:\HDS_testdb_DATA\Broker1\Broker_1.ndf	broker_fg	31795200 KB	Unlimited	10240 KB	data only
7	Broker2	7	H:\HDS_testdb_DATA\Broker2\Broker_2.ndf	broker_fg	31784960 KB	Unlimited	10240 KB	data only
8	Broker3	8	J:\HDS_testdb_DATA\Broker3\Broker_3.ndf	broker_fg	31784960 KB	Unlimited	10240 KB	data only
9	Market1	9	J:\HDS_testdb_DATA\Market1\Market_1.ndf	market_fg	1024000 KB	Unlimited	10240 KB	data only
10	Market2	10	I:\HDS_testdb_DATA\Market2\Market_2.ndf	market_fg	1024000 KB	Unlimited	10240 KB	data only
11	Market3	11	H:\HDS_testdb_DATA\Market3\Market_3.ndf	market_fg	1024000 KB	Unlimited	10240 KB	data only
12	Customer1	12	H:\HDS_testdb_DATA\Customer1\Customer_1.ndf	customer_fg	6748160 KB	Unlimited	10240 KB	data only
13	Customer2	13	J:\HDS_testdb_DATA\Customer2\Customer_2.ndf	customer_fg	6748160 KB	Unlimited	10240 KB	data only
14	Customer3	14	I:\HDS_testdb_DATA\Customer3\Customer_3.ndf	customer_fg	6748160 KB	Unlimited	10240 KB	data only

This information can help determine what files, filegroups and file identifiers are used by the selected database. For this particular output, note the size of the files and their growth in kilobytes. Check to see that all database files for a filegroup are about the same size and grow at the same rate.

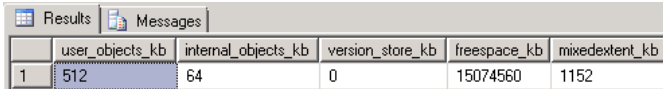
Monitoring tempdb Usage

Use the following SQL code to monitor the number of user objects, internal objects, version stores and free space left in the tempdb file. This code uses the database DMVs.

```
SELECT SUM (user_obj ect_reserved_page_count) *8 as user_obj ects_kb
, SUM (i nternal_obj ect_reserved_page_count) *8 as i nternal_obj ects_kb
, SUM (versi on_store_reserved_page_count) *8 as versi on_store_kb
, SUM (unali cated_extent_page_count) *8 as freespace_kb
, SUM (mi xed_extent_page_count) *8 as mi xedextent_kb
From sys.dm_db_fi l e_space_usage
WHERE database_i d = 2
```

Running this SQL code produces the results shown in Figure 7. This information is useful for determining the usage of tempdb (database_i d = 2 uniquely identifies tempdb).

Figure 7. tempdb Size Information



	user_objects_kb	internal_objects_kb	version_store_kb	freespace_kb	mixedextent_kb
1	512	64	0	15074560	1152

This information can help determine the usage of tempdb. A high number of user or internal objects might indicate heavy usage of variables, temporary tables, sort objects and other objects signifying heavy use of tempdb. Also note the free space remaining for tempdb.

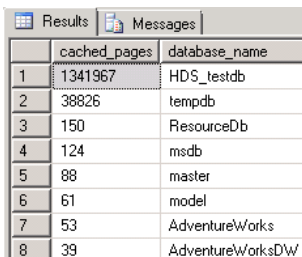
Monitoring Cache Usage

The following SQL code provides the number of data pages each database has in the buffer cache.

```
SELECT    count(*) AS cached_pages
          , CASE database_id
            WHEN 32767 THEN 'ResourceDb'
            ELSE db_name(database_id)
            END
          AS database_name
FROM      sys.dm_os_buffer_descriptors
GROUP BY db_name(database_id)
         , database_id
ORDER BY cached_pages DESC
```

Running this SQL code produces the results shown in Figure 8.

Figure 8. cached_pages Output



	cached_pages	database_name
1	1341967	HDS_testdb
2	38826	tempdb
3	150	ResourceDb
4	124	msdb
5	88	master
6	61	model
7	53	AdventureWorks
8	39	AdventureWorksDW

This information can help DBAs determine which databases are consuming the most cache at a given time.

Monitoring Latch Waits

I/O bottlenecks can sometimes be identified by examining latch waits. A latch wait is a short-term, lightweight synchronization object that accounts for the physical I/O waits when accessing a page for reading or writing and the page is not available in the buffer pool. When the page is not found in the buffer pool, an asynchronous I/O is posted and then the status of the I/O is checked. If I/O is complete, the worker process proceeds normally. Otherwise, it waits on PAGEI OLATCH_EX or PAGEI OLATCH_SH (exclusive or shared respectively), depending upon the type of request.

The following SQL query provides I/O wait latch statistics.

```
SELECT    wait_type
          , waiting_tasks_count
          , wait_time_ms
FROM      sys.dm_os_wait_stats
WHERE     wait_type like 'PAGEI OLATCH%'
ORDER BY wait_type
```

Running this SQL code produces the results in Figure 9. This DMV table, sys.dm_os_wait_stats, lists all wait latch types. This example uses the WHERE clause to limit the search to only page latch types.

Figure 9. wait_type and wait_time Output

	wait_type	waiting_tasks_count	wait_time_ms
1	PAGEIOLATCH_DT	0	0
2	PAGEIOLATCH_EX	19	303
3	PAGEIOLATCH_KP	0	0
4	PAGEIOLATCH_NL	0	0
5	PAGEIOLATCH_SH	138	4961
6	PAGEIOLATCH_UP	24	249

Note that this information is cumulative and gathered from the time the server starts. To clear these values, either restart the SQL Server instance or make a call to the following ODBC function:

```
DBCC SQLPERF (' sys.dm_os_wait_stats' , CLEAR)
```

Monitoring I/O Writes and Sizes

When the I/O of one or more databases is suspected to be the cause of a performance issue, the following SQL code can be executed to determine which database or databases and files have the largest I/O and longest average response time.

```
SELECT db_name(database_id) AS Database_Name
, file_id
, CAST(num_of_reads AS BIGINT) AS [total_num_reads]
, CAST(num_of_writes AS BIGINT) AS [total_num_writes]
, CAST(io_stall_read_ms AS BIGINT)
/ CAST(CASE WHEN num_of_reads=0
THEN 1
ELSE num_of_reads
END AS BIGINT) AS [avg_read_stall]
, CAST(io_stall_write_ms AS BIGINT)
/ CAST(CASE WHEN num_of_writes=0
THEN 1
ELSE num_of_writes
END AS BIGINT) AS [avg_write_stall]
, CAST(num_of_bytes_read AS BIGINT) AS [total_bytes_read]
, CAST(num_of_bytes_written AS BIGINT) AS [total_bytes_written]
, CAST(num_of_bytes_read AS BIGINT)
/ CAST(CASE WHEN num_of_reads=0
THEN 1
ELSE num_of_reads
END AS BIGINT)
/ 1024 AS [avg_read_size_Kbytes]
, CAST(num_of_bytes_written AS BIGINT)
/ CAST(CASE WHEN num_of_writes=0
THEN 1
ELSE num_of_writes
END AS BIGINT)
/ 1024 AS [avg_write_size_Kbytes]
FROM sys.dm_io_virtual_file_stats(NULL, NULL)
```

Running this SQL code produces the results in Figure 10. This DMV table lists a number of I/O related totals and averages for all databases in a SQL instance. This can help determine whether a specific database might be causing an I/O problem for the entire SQL environment.

Figure 10. I/O Writes and Sizes Output

	Database_Name	file_id	total_num_reads	total_num_writes	avg_read_stall	avg_write_stall	total_bytes_read	total_bytes_written	avg_read_size_Kbytes	avg_write_size_Kbytes
1	master	1	38	1	15	20	2367488	8192	60	8
2	master	2	13	19	13	19	475136	81920	35	4
3	tempdb	1	101	15	4	0	6438912	172032	62	11
4	tempdb	2	13	132	0	0	475136	7382016	35	54
5	tempdb	3	14	101	1	0	229376	835584	16	8
6	tempdb	4	14	101	0	0	229376	843776	16	8
7	model	1	45	2	6	19	2777088	16384	60	8
8	model	2	9	7	6	12	458752	36864	49	5
9	msdb	1	34	1	17	82	2056192	8192	59	8
10	msdb	2	13	4	16	33	475136	20480	35	5
11	AdventureWorksDWH	1	19	1	15	95	1073152	8192	55	8
12	AdventureWorksDWH	2	9	5	17	52	458752	24576	49	4

Note that this information is cumulative and gathered from the time the server starts. To clear these values, either restart the SQL Server instance or make a call to the following ODBC function:

```
DBCC SQLPERF (' sys.dm_os_waits_stats', CLEAR)
```

This function resets all wait statistic information, not just the pagelatch information.

For more information about wait types, see [Microsoft's Help and Support Knowledge Base article 822101](#) and the Microsoft Knowledge Base article [SQL Server 2005 Waits and Queues](#).

Monitoring I/O Pending

The following SQL query provides I/O pending for each database in a SQL server instance. The DMF takes two parameters, database number and file number, both of which can be null, and produces very granular information on database reads, writes, I/O stalls, number of bytes on disk and more.

```
SELECT database_id
       , file_id
       , io_stall
       , io_pending_ms_ticks
       , scheduler_address
FROM sys.dm_io_virtual_file_stats(NULL, NULL) stats
     , sys.dm_io_pending_io_requests as requests
WHERE stats.file_handle = requests.io_handle
```

As shown in Figure 11, an empty result data set is an indication that the databases and queries are well constructed or the databases are not currently very busy.

Figure 11. I/O stall and pending Output

database_id	file_id	io_stall	io_pending_ms_ticks	scheduler_address



Corporate Headquarters 750 Central Expressway, Santa Clara, California 95050-2627 USA
Contact Information: + 1 408 970 1000 www.hds.com / info@hds.com

Asia Pacific and Americas 750 Central Expressway, Santa Clara, California 95050-2627 USA □
Contact Information: + 1 408 970 1000 www.hds.com / info@hds.com

Europe Headquarters Sefton Park, Stoke Poges, Buckinghamshire SL2 4HD United Kingdom
Contact Information: + 44 (0) 1753 618000 www.hds.com / info.uk@hds.com

Hitachi is a registered trademark of Hitachi, Ltd. in the United States and others countries. Hitachi Data Systems is a registered trademark and service mark of Hitachi, Ltd. in the United States and other countries.

IBM , ESCON and FICON are registered trademarks of the IBM corporation.

All other trademarks, service marks and company names mentioned in this document are properties of their respective owners.

Notice: This document is for information purposes only, and does not set forth any warranty, expressed or implied, concerning any equipment or service offered or to be offered by Hitachi Data Systems. This document describes some capabilities that are conditioned on a maintenance contract with Hitachi Data Systems being in effect, and that may be configuration dependent, and features that may not be currently available. Contact your local Hitachi Data Systems sales office for information on feature and product availability.

© Hitachi Data Systems Corporation 2009. All Rights Reserved
AS-002-01 June 2009