Performance Tuning Guidelines for Windows Server 2008

June 9, 2008

Abstract

This guide describes important tuning parameters and settings that can result in improved performance for the Windows Server® 2008 operating system. Each setting and its potential effect are described to help you make an informed judgment about its relevance to your system, workload, and performance goals.

This information applies to the Windows Server 2008 operating system.

The current version of this guide is maintained on the Web at:   
 <http://www.microsoft.com/whdc/system/sysperf/Perf_tun_srv.mspx>

Feedback: Please tell us if this paper was useful to you. Submit comments at:  
 <http://go.microsoft.com/fwlink/?LinkId=102585>

References and resources discussed here are listed at the end of this guide.

**Disclaimer:** This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place or event is intended or should be inferred.

© 2007–2008 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, MS-DOS, MSDN, SQL Server, Win-32, Windows, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Document History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Change |  |  |  |
| May 27, 2008 | Added “Power Guidelines” under Server Hardware section and added “Performance Tuning for Virtualization Servers” section. | | | |
| October 16, 2007 | Added “Performance Tuning for Terminal Server” and “Performance Tuning for Terminal Server Gateway” sections. | | | |
| August 31, 2007 | First publication | | | |

Contents

[Introduction 6](#_Toc200504769)

[In This Guide 6](#_Toc200504770)

[Performance Tuning for Server Hardware 7](#_Toc200504771)

[Power Guidelines 9](#_Toc200504772)

[Interrupt Affinity 10](#_Toc200504773)

[Performance Tuning for Networking Subsystem 10](#_Toc200504774)

[Choosing a Network Adapter 12](#_Toc200504775)

[Offload Capabilities 12](#_Toc200504776)

[Receive-Side Scaling (RSS) 12](#_Toc200504777)

[Message-Signaled Interrupts (MSI/MSI-X) 12](#_Toc200504778)

[Network Adapter Resources 13](#_Toc200504779)

[Interrupt Moderation 13](#_Toc200504780)

[Tuning the Network Adapter 13](#_Toc200504781)

[Enabling Offload Features 13](#_Toc200504782)

[Increasing Network Adapter Resources 14](#_Toc200504783)

[Enabling Interrupt Moderation 14](#_Toc200504784)

[Binding Each Adapter to a CPU 14](#_Toc200504785)

[TCP Receive Window Auto-Tuning 14](#_Toc200504786)

[TCP Parameters 15](#_Toc200504787)

[Network-Related Performance Counters 15](#_Toc200504788)

[Performance Tuning for Storage Subsystem 16](#_Toc200504789)

[Choosing Storage 17](#_Toc200504790)

[Estimating the Amount of Data to Be Stored 18](#_Toc200504791)

[Choosing a Storage Array Selection 19](#_Toc200504792)

[Hardware RAID Levels 19](#_Toc200504793)

[Choosing the RAID Level 22](#_Toc200504794)

[Selecting a Stripe Unit Size 26](#_Toc200504795)

[Determining the Volume Layout 26](#_Toc200504796)

[Storage-Related Parameters 27](#_Toc200504797)

[NumberOfRequests 27](#_Toc200504798)

[I/O Priorities 27](#_Toc200504799)

[Storage-Related Performance Counters 28](#_Toc200504800)

[Logical Disk and Physical Disk 28](#_Toc200504801)

[Processor 29](#_Toc200504802)

[Power Protection and Advanced Performance Option 30](#_Toc200504803)

[Block Alignment (DISKPART) 30](#_Toc200504804)

[Solid-State and Hybrid Drives 30](#_Toc200504805)

[Response Times 31](#_Toc200504806)

[Queue Lengths 32](#_Toc200504807)

[Performance Tuning for Web Servers 33](#_Toc200504808)

[Selecting the Proper Hardware for Performance 33](#_Toc200504809)

[Operating System Practices 33](#_Toc200504810)

[Tuning IIS 7.0 33](#_Toc200504811)

[Kernel-Mode Tunings 35](#_Toc200504812)

[Cache Management Settings 35](#_Toc200504813)

[Request and Connection Management Settings 36](#_Toc200504814)

[User-Mode Settings 37](#_Toc200504815)

[User-Mode Cache Behavior Settings 37](#_Toc200504816)

[Compression Behavior Settings 37](#_Toc200504817)

[Tuning the Default Document List 38](#_Toc200504818)

[Central Binary Logging 39](#_Toc200504819)

[Application and Site Tunings 40](#_Toc200504820)

[Managing IIS 7.0 Modules 41](#_Toc200504821)

[Classic ASP Settings 41](#_Toc200504822)

[ASP.NET Concurrency Setting 42](#_Toc200504823)

[Worker Process and Recycling Options 42](#_Toc200504824)

[Secure Sockets Layer Tuning Parameters 43](#_Toc200504825)

[ISAPI 43](#_Toc200504826)

[Managed Code Tuning Guidelines 43](#_Toc200504827)

[Other Issues that Affect IIS Performance 44](#_Toc200504828)

[NTFS File System Setting 44](#_Toc200504829)

[Networking Subsystem Performance Settings for IIS 44](#_Toc200504830)

[Performance Tuning for File Servers 44](#_Toc200504831)

[Selecting the Proper Hardware for Performance 44](#_Toc200504832)

[Server Message Block Model 45](#_Toc200504833)

[Configuration Considerations 45](#_Toc200504834)

[General Tuning Parameters for Servers 46](#_Toc200504835)

[General Tuning Parameters for Client Computers 47](#_Toc200504836)

[Performance Tuning for Active Directory Servers 47](#_Toc200504837)

[Considerations for Read-Heavy Scenarios 48](#_Toc200504838)

[Considerations for Write-Heavy Scenarios 49](#_Toc200504839)

[Using Indexing to Increase Query Performance 49](#_Toc200504840)

[Optimizing Trust Paths 49](#_Toc200504841)

[Active Directory Performance Counters 50](#_Toc200504842)

[Performance Tuning for Terminal Server 51](#_Toc200504843)

[Selecting the Proper Hardware for Performance 51](#_Toc200504844)

[CPU Configuration 51](#_Toc200504845)

[Processor Architecture 51](#_Toc200504846)

[Memory Configuration 52](#_Toc200504847)

[Disk 52](#_Toc200504848)

[Network 53](#_Toc200504849)

[Tuning Applications for Terminal Server 53](#_Toc200504850)

[Terminal Server Tuning Parameters 54](#_Toc200504851)

[Pagefile 54](#_Toc200504852)

[Antivirus and Antispyware 54](#_Toc200504853)

[Task Scheduler 54](#_Toc200504854)

[Desktop Notification Icons 55](#_Toc200504855)

[Client Experience Settings 56](#_Toc200504856)

[Desktop Size 57](#_Toc200504857)

[Windows System Resource Manager 57](#_Toc200504858)

[Performance Tuning for Terminal Server Gateway 57](#_Toc200504859)

[Monitoring and Data Collection 58](#_Toc200504860)

[Performance Tuning for Virtualization Servers 58](#_Toc200504861)

[Terminology 59](#_Toc200504862)

[Hyper‑V Architecture 60](#_Toc200504863)

[Server Configuration 61](#_Toc200504864)

[Hardware Selection 61](#_Toc200504865)

[Server Core Installation Option 62](#_Toc200504866)

[Dedicated Server Role 62](#_Toc200504867)

[Guest Operating Systems 62](#_Toc200504868)

[CPU Statistics 63](#_Toc200504869)

[Processor Performance 63](#_Toc200504870)

[Integration Services 63](#_Toc200504871)

[Enlightened Guests 63](#_Toc200504872)

[Virtual Processors 63](#_Toc200504873)

[Background Activity 64](#_Toc200504874)

[Weights and Reserves 64](#_Toc200504875)

[Memory Performance 64](#_Toc200504876)

[Enlightened Guests 65](#_Toc200504877)

[Correct Memory Sizing 65](#_Toc200504878)

[Storage I/O Performance 65](#_Toc200504879)

[Synthetic SCSI Controller 65](#_Toc200504880)

[Virtual Hard Disk Types 66](#_Toc200504881)

[Passthrough Disks 66](#_Toc200504882)

[Disabling File Last Access Time Check 67](#_Toc200504883)

[Physical Disk Topology 67](#_Toc200504884)

[I/O Balancer Controls 67](#_Toc200504885)

[Network I/O Performance 68](#_Toc200504886)

[Synthetic Network Adapter 68](#_Toc200504887)

[Offload Hardware 68](#_Toc200504888)

[Network Switch Topology 68](#_Toc200504889)

[Interrupt Affinity 69](#_Toc200504890)

[VLAN Performance 69](#_Toc200504891)

[Performance Tuning for File Server Workload (NetBench) 69](#_Toc200504892)

[Registry Tuning Parameters for Servers 69](#_Toc200504893)

[Registry Tuning Parameters for Client Computers 70](#_Toc200504894)

[Performance Tuning for Network Workload (NTttcp) 70](#_Toc200504895)

[Tuning for NTttcp 70](#_Toc200504896)

[Network Adapter 71](#_Toc200504897)

[TCP/IP Window Size 71](#_Toc200504898)

[Receive-Side Scaling (RSS) 71](#_Toc200504899)

[Tuning for Chariot 72](#_Toc200504900)

[Performance Tuning for Terminal Server Knowledge Worker Workload 72](#_Toc200504901)

[Recommended Tunings on the Server 73](#_Toc200504902)

[Monitoring and Data Collection 75](#_Toc200504903)

[Performance Tuning for SAP Sales and Distribution Two-Tier Workload 75](#_Toc200504904)

[Operating System Tunings on the Server 76](#_Toc200504905)

[Tunings on the Database Server 76](#_Toc200504906)

[Tunings on the SAP Application Server 77](#_Toc200504907)

[Monitoring and Data Collection 77](#_Toc200504908)

[Resources 78](#_Toc200504909)

# Introduction

Windows Server® 2008 should perform very well out of the box for most customer workloads. Optimal out-of-the-box performance was a major goal for this release and influenced how Microsoft designed a new, dynamically tuned networking subsystem that incorporates both IPv4 and IPv6 protocols and improved file sharing through Server Message Block (SMB) 2.0. However, you can further tune the server settings and obtain incremental performance gains, especially when the nature of the workload varies little over time.

The most effective tuning changes consider the hardware, the workload, and the performance goals. This guide describes important tuning considerations and settings that can result in improved performance. Each setting and its potential effect are described to help you make an informed judgment about its relevance to your system, workload, and performance goals.

**Note**: Registry settings and tuning parameters have changed significantly from Windows Server 2003 to Windows Server 2008. Remember this as you tune your server—using earlier or out-of-date tuning guidelines might produce unexpected results.

As always, be careful when you directly manipulate the registry. If you must edit the registry, back it up first.

# In This Guide

This guide contains key performance recommendations for the following components:

* [Server Hardware](#_Server_Hardware)
* [Networking Subsystem](#_Performance_Tuning_for)
* [Storage Subsystem](#_Performance_Tuning_for_1)

This guide also contains performance tuning considerations for the following server roles:

* [Web Servers](#_Performance_Tuning_for_IIS_6.0)
* [File Servers](#_Performance_Tuning_for_2)
* [Active Directory Servers](#_Performance_Tuning_for_3)
* [Terminal Servers](#_Performance_Tuning_for_5)
* [Terminal Server Gateway](#_Performance_Tuning_for_4)
* [Virtualization Server (Hyper‑V)](#_Performance_Tuning_for_6)
* [File Server Workload](#_Performance_Tuning_for_6)
* [Networking Workload](#_Performance_Tuning_for_7)
* [Terminal Server Knowledge Worker Workload](#_Terminal_Server_Knowledge)
* [SAP Sales and Distribution Two-Tier Workload](#_SAP_Sales_and)

# Performance Tuning for Server Hardware

It is important to select the proper hardware to satisfy the expected performance goals. Hardware bottlenecks limit the effectiveness of software tuning. This section provides guidelines for laying a good foundation for the role that a server will play. Later sections provide tuning guidelines that are specific to a server role and include diagnostic techniques for isolating and identifying performance bottlenecks for certain server roles.

Table 1 provides important considerations that you should use when you choose the server hardware. Following these guidelines can help remove artificial performance bottlenecks that might impede the server’s performance.

Table 1. Server Hardware Recommendations

|  |  |
| --- | --- |
| Component | Recommendation |
| Processors | When the option is available, choose 64-bit processors because of the benefit of additional address space.  Research data shows that two CPUs are not as fast as one CPU that is twice as fast. Because it is not always possible to obtain a CPU that is twice as fast, doubling the number of CPUs is preferred, but does not guarantee twice the performance.  It is important to match and scale the memory and I/O subsystem with the CPU power and vice versa.  Do not compare CPU frequencies across manufacturers and generations because the comparison can be a misleading indicator of speed. |
| Cache | Choose large L2 or L3 processor caches. The larger caches generally provide better performance and often play a bigger role than raw CPU frequency. |
| Memory (RAM) and Paging Storage | Increase the RAM to match your memory needs. When your computer runs low on memory and needs more immediately, modern operating systems use hard disk space to supplement system RAM through a procedure called paging. Too much paging degrades overall system performance.  You can optimize paging by using the following guidelines for pagefile placement:  Place the pagefile and operating system files on separate physical disk drives.  Place the pagefile on a drive that is not fault-tolerant. Note that, if the disk dies, a system crash is highly possible. If you place the pagefile on a fault-tolerant drive, remember that some fault-tolerant systems experience slow data writes because they write data to multiple locations.  Use multiple disks or a disk array if additional disk bandwidth is needed for paging. Do not place multiple pagefiles on different partitions of the same physical disk drive. |
| Peripheral Bus | To avoid bus speed limitations, use either PCI-X or PCIe x8 and higher slots for Gigabit Ethernet adapters. |
| Disks | Higher rotational speeds reduce random request service times (~2 ms on average when you compare 7,200- and 15,000-RPM drives) and increase sequential request bandwidth.  The latest generation of 2.5-inch enterprise-class disks can service a significantly larger number of random requests per second compared to 3.5-inch drives.  Store “hot” data near the “beginning” of a disk because this corresponds to the outermost (fastest) tracks.  Consolidating small drives into fewer high-capacity drives can easily reduce overall storage performance. Fewer spindles mean reduced request service concurrency and therefore potentially lower throughput and longer response times (depending on the workload intensity). |

Table 2 lists the recommended settings for choosing networking and storage adapters in a high-performing server environment. These settings can help keep your networking or storage hardware from being the bottleneck when they are under heavy load.

Table 2. Networking and Storage Adapter Recommendations

|  |  |
| --- | --- |
| Recommen-dation | Description |
| WHQL certified | The adapter has passed the Windows® Hardware Quality Labs (WHQL) certification test suite. |
| 64-bit capability | Adapters that are 64-bit capable can perform direct memory access (DMA) operations to and from high physical memory locations (greater than 4 GB). If the driver does not support DMA greater than 4 GB, the system double-buffers the I/O to a physical address space of less than 4 GB. |
| Copper and fiber (glass) adapters | Copper adapters generally have the same performance as their fiber counterparts, and both copper and fiber are available on some Fibre Channel adapters. Certain environments are better suited to copper adapters, whereas other environments are better suited to fiber adapters. |
| Dual- or quad-port adapters | Multiport adapters are useful for servers that have limited PCI slots.  To address SCSI limitations on the number of disks that can be connected to a SCSI bus, some adapters provide two or four SCSI buses on a single adapter card. Fibre Channel disks generally have no limits to the number of disks that are connected to an adapter unless they are hidden behind a SCSI interface.  Serial Attached SCSI (SAS) and Serial ATA (SATA) adapters also have a limited number of connections because of the serial nature of the protocols, but more attached disks are possible by using switches.  Network adapters have this feature for load-balancing or failover scenarios. Using two single-port network adapters usually yields better performance than using a single dual-port network adapter for the same workload.  PCI bus limitation can be a major factor in limiting performance for multiport adapters. Therefore, it is important to consider placing them in a high-performing PCI slot that provides enough bandwidth. Generally, PCI‑E adapters provide more bandwidth than PCI‑X adapters. |
| Interrupt moderation | Some adapters can moderate how frequently they interrupt the host processors to indicate activity (or its completion). Moderating interrupts can often result in reduced CPU load on the host but, unless interrupt moderation is performed intelligently, the CPU savings might increase latency. |
| Offload capability and other advanced features such as message-signaled interrupt (MSI)-X | Offload-capable adapters offer CPU savings that translate into improved performance. For more information, see “[Choosing a Network Adapter](#_Choosing_a_Network)” later in this guide. |
| Dynamic interrupt and deferred procedure call (DPC) redirection | Windows Server 2008 has new functionality that enables PCI-E storage adapters to dynamically redirect interrupts and DPCs. This capability, originally called “NUMA I/O,” can help any multiprocessor system by improving workload partitioning, cache hit rates, and on-board hardware interconnect usage for I/O-intensive workloads. At Windows Server 2008 RTM, no adapters on the market had this capability, but several manufacturers were developing adapters to take advantage of this performance feature. |

## Power Guidelines

Although this guide focuses on how to obtain the best performance from Windows Server 2008, the increasing importance of power efficiency must also be recognized in enterprise and data center environments. High performance and low power usage are often conflicting goals, but by carefully selecting server components you can determine the correct balance between them. Table 3 contains guidelines for power characteristics and capabilities of server hardware components.

Table 3. Server Hardware Power Savings Recommendations

|  |  |
| --- | --- |
| Component | Recommendation |
| Processors | Higher frequencies in a specific processor family cause increased power consumption when the processors are under heavy load. Also, processor families usually include low-power versions. Newer generations of processors expose more power states for the Windows power management algorithms, which enables better power management at all levels of performance. |
| Memory (RAM) | Memory consumes an increasing part of system power. Many factors affect the power consumption of a memory “stick” such as memory technology, error correction code (ECC), frequency, capacity, density, and number of ranks. Therefore, it is best to compare expected power consumption ratings before purchasing large quantities of memory. Low-power (“green”) memory is now available, but a performance or monetary trade-off must be considered. If paging is required, then the power cost of the paging disks should also be considered. |
| Disks | Higher RPM means increased power consumption. Also, new 2.5-inch drives consume less than half the power of older 3.5-inch drives. More information about the power cost for different RAID configurations is found in “[Performance Tuning for Storage Subsystem](#_Performance_Tuning_for_1)” later in this guide. |
| Network and storage adapters | Some adapters decrease power consumption during idle periods. This becomes a more important consideration for 10‑Gb networking and high-bandwidth storage links. |

The default power plan for Windows Server 2008 is Balanced, which tries to keep performance high while it saves power whenever possible. The other predefined plans are Low Power and High Performance, both of which are heavily weighted to different goals. But server BIOS settings can prevent Windows from accomplishing any of these goals, so make sure that you check whether power management by the operating system or by the hardware is a BIOS option. Windows Server performance lab tests show that Windows power management works very well when it is compared to hardware-managed power management on enterprise servers, so the operating system–managed setting is preferred. However, the most important guideline is to make sure that the BIOS settings on a specific server are well understood so that the administrator knows if the Windows power setting controls (including the High Performance plan) are actually usable.

## Interrupt Affinity

Interrupt affinity refers to the binding of interrupts from a specific device to one or more specific processors in a multiprocessor server. The binding forces interrupt processing to run on the specified processor or processors, unless the device specifies otherwise. For some scenarios, such as a file server, the network connections and file server sessions remain on the same network adapter. In those scenarios, binding interrupts from the network adapter to a processor allows for processing incoming packets (SMB requests and data) on a specific set of processors, which improves locality and scalability.

The Interrupt-Affinity Filter tool (IntFiltr) lets you change the CPU affinity of the interrupt service routine (ISR). The tool runs on most servers that run Windows Server 2008, regardless of what processor or interrupt controller is used. However, on some systems with more than eight logical processors or for devices that use MSI or MSI-X, the tool is limited by the Advanced Programmable Interrupt Controller (APIC) protocol. The [Interrupt-Affinity Policy](http://www.microsoft.com/whdc/system/sysperf/IntPolicy.mspx) tool does not encounter this issue because it sets the CPU affinity through the affinity policy of a device.

You can use this tool to direct any device's ISR to a specific processor or set of processors (instead of sending interrupts to any of the CPUs in the system). Note that different devices can have different interrupt affinity settings. For IntFiltr to work on some systems, you must set the MAXPROCSPERCLUSTER=0 boot parameter. On some systems, directing the ISR to a processor on a different nonuniform memory access (NUMA) node can cause performance issues.

# Performance Tuning for Networking Subsystem

Figure 1 shows the network architecture, which covers many components, interfaces, and protocols. The following sections discuss tuning guidelines for some components of server workloads.

Figure 1. Network Stack Components

***WMS***

***HTTP.SYS***

***TCP/IP***

***DNS***

***IIS***

***AFD.SYS***

***NIC Driver***

***User-Mode Applications***

***System Drivers***

***Protocol Stack***

***NDIS***

***Network Interface***

***UDP/IP***

***VPN***

***NDIS***

The network architecture is layered, and the layers can be broadly divided into the following sections:

* The network driver and Network Driver Interface Specification (NDIS).

These are the lowest layers. NDIS exposes interfaces for the driver below it and for the layers above it such as TCP/IP.

* The protocol stack.

This implements protocols such as TCP/IP and UDP/IP. These layers expose the transport layer interface for layers above them.

* System drivers.

These are typically transport data interface extension (TDX) or Winsock Kernel (WSK) clients and expose interfaces to user-mode applications. The WSK interface is a new feature for Windows Server 2008 and Windows Vista® that is exposed by Afd.sys. The interface improves performance by eliminating the switching between user mode and kernel modes.

* User-mode applications.

These are typically Microsoft solutions or custom applications.

Tuning for network-intensive workloads can involve each layer. The following sections describe some tuning changes.

## Choosing a Network Adapter

Network-intensive applications need high-performance network adapters. This section covers some considerations for choosing network adapters.

### Offload Capabilities

Offloading tasks can help reduce CPU usage on the server, which improves overall system performance. The Microsoft networking stack can offload one or more tasks to a network adapter that has the appropriate task-offload capabilities. Table 4 provides more details about each offload.

Table 4. Offload Capabilities for Network Adapters

| **Offload type** | **Description** |
| --- | --- |
| Checksum calculation | The networking stack can offload the calculation and validation of both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) checksums on sends and receives. It can also offload the calculation and validation of both IPv4 and IPv6 checksums on sends and receives. |
| IP security authentication and encryption | The TCP/IP transport can offload the calculation and validation of encrypted checksums for authentication headers and Encapsulating Security Payloads (ESPs). The TCP/IP transport can also offload the encryption and decryption of ESPs. |
| Segmentation of large TCP packets | The TCP/IP transport supports Giant Send Offload (GSO). With GSO, also known as LSOv2, the TCP/IP transport can offload the segmentation of large TCP packets. |
| TCP stack | The TCP offload engine (TOE) enables a network adapter that has the appropriate capabilities to offloadthe entire network stack. |

### Receive-Side Scaling (RSS)

On systems with Pentium 4 and later processors, the scheduling for processing networking I/O within the context of an ISR is routed to the same processor. This behavior differs from that of earlier processors in which interrupts from a device are rotated to all processors. The result is a scalability limitation for multiprocessor servers that host a single network adapter that is governed by the processing power of a single CPU. With RSS, the network driver together with the network card distributes incoming packets among processors so that packets that belong to the same TCP connection are on the same processor, which preserves ordering. This helps improve scalability for scenarios such as Web servers, in which a machine accepts many connections that originate from different source addresses and ports. Research shows that distributing packets that belong to TCP connections across hyperthreading processors degrades performance. Therefore, only physical processors accept RSS traffic. For more information about RSS, see “[Scalable Networking: Eliminating the Receive Processing Bottleneck—Introducing RSS](http://72.14.253.104/search?q=cache:QTv6ttmAbQUJ:download.microsoft.com/download/5/D/6/5D6EAF2B-7DDF-476B-93DC-7CF0072878E6/NDIS_RSS.doc+Receive+Side+Scaling+hash+2-tuple&hl=en&ct=clnk&cd=1&gl=us).”

### Message-Signaled Interrupts (MSI/MSI-X)

The ability to target processors with interrupts that are coupled with RSS dedicates a processor to servicing interrupts and DPCs that belong to the same TCP connection. This preserves the cache locality of TCP structures and greatly improves performance.

### Network Adapter Resources

Several network adapters let the administrator manually configure resources by using the **Advanced Networking** tab for the adapter. Receive buffers and send buffers are among the parameters that can be set. Only a few network adapters actively manage their resources, so setting parameters for these network adapters is unnecessary.

### Interrupt Moderation

To control interrupt moderation, some network adapters expose either different interrupt moderation levels, buffer coalescing parameters (sometimes separately for send and receive buffers), or both. You should consider buffer coalescing or batching when the network adapter does not perform interrupt moderation.

Table 5 provides a guideline of which high-performance features improve performance in terms of throughput, latency, or scalability for some server roles.

Table 5. Benefits from Network Adapter Features for Different Server Roles

| **Server role** | **Checksum offload** | **Segmentation offload** | **TCP offload engine (TOE)** | **Receive-side scaling (RSS)** |
| --- | --- | --- | --- | --- |
| File server | X | X | X |  |
| Web server | X | X | X | X |
| Mail server (short-lived connections) | X |  |  | X |
| Database server | X | X | X | X |
| FTP server | X | X | X |  |
| Media server | X |  | X | X |

**Disclaimer**: The recommendations in Table 5 are intended to serve as guidance only for choosing the most suitable technology for specific server roles under a deterministic traffic pattern. User experience can be different, depending on workload characteristics and the hardware that is used.

If your hardware supports TOE, then you must enable that option in the operating system to benefit from the hardware’s capability. You can enable TOE by running the following:

netsh int tcp set global chimney = enabled

## Tuning the Network Adapter

You can optimize network throughput and resource usage by using network adapter tunings (when they are available and exposed by the network adapter). Remember that the correct set of tunings depends on the network adapter, the workload, the host computer resources, and your performance goals.

### Enabling Offload Features

Turning on network adapter offload features is usually beneficial. Sometimes, however, the network adapter is not powerful enough to handle the offload capabilities at high throughput. For example, enabling segmentation offload can reduce the maximum sustainable throughput on some network adapters because of limited hardware resources. However, if the reduced throughput is not expected to be a limitation, you should enable offload capabilities even for such network adapters. Note that some network adapters require offload features to be independently enabled for send and receive paths.

### Increasing Network Adapter Resources

For network adapters that allow for the manual configuration of resources such as receive and send buffers, you should increase the allocated resources. Some network adapters set their receive buffers low to conserve allocated memory from the host. The low value results in dropped packets and decreased performance. Therefore, for receive-intensive scenarios, we recommend that you increase receive buffer value to the maximum. If the adapter does not expose manual resource configuration, then it either dynamically configures the resources or is statically set to a fixed value that cannot be changed.

### Enabling Interrupt Moderation

To control interrupt moderation, some network adapters expose different interrupt moderation levels, buffer coalescing parameters (sometimes separately for send and receive buffers), or both. You should consider interrupt moderation for CPU-bound workloads and consider the trade‑off between the host CPU savings and latency versus the increased host CPU savings because of more interrupts and less latency. If the network adapter does not perform interrupt moderation but does expose buffer coalescing, then increasing the number of coalesced buffers allows for more buffers per send or receive, which improves performance.

### Binding Each Adapter to a CPU

The method to use depends on the number of network adapters, the number of CPUs, and the number of ports per network adapter. Important factors are the type of workload and the distribution of the interrupts across the CPUs. For a workload such as a Web server that has several networking adapters, partition the adapters on a processor basis to isolate the interrupts that the adapters generate.

## TCP Receive Window Auto-Tuning

One of the most significant changes to the TCP stack for this release is TCP receive window auto-tuning, which can affect existing network infrastructure demands. Previously, the network stack used a fixed-size receive-side window that limited the overall potential throughput for connections. You can calculate the total throughput of a single connection when you use this fixed size default as:

Total achievable throughput in bytes = TCP window \* (1 / connection latency)

For example, the total achievable throughput is only 51 Mbps on a 1-GB connection with a 10-ms latency (a reasonable value for a large corporate network infrastructure). With auto-tuning, however, the receive-side window is adjustable and can grow to meet the demands of the sender. It is entirely possible for a connection to achieve a full line rate of a 1‑GB connection. Network usage scenarios that might have been limited in the past by the total achievable throughput of TCP connections now can fully use the network.

The remote file copy is a common network usage scenario that is likely to increase demand on the infrastructure because of this change. Many improvements have been made to the underlying operating system support for remote file copy that now let large file copies perform at disk I/O speeds. If many concurrent remote large file copies are typical within your network environment, your network infrastructure might be taxed by the significant increase in network usage by each file copy operation.

**Windows Filtering Platform**

The Windows Filtering Platform (WFP) that was introduced in Windows Vista and Windows Server 2008 provides APIs to third-party independent software vendors (ISVs) to create packet processing filters. Examples include firewall and antivirus software. Note that a poorly written WFP filter significantly decreases a server’s networking performance. For more information about WFP, see “[Windows Filtering Platform](http://www.microsoft.com/whdc/device/network/WFP.mspx).”

## TCP Parameters

The following keywords, which for Windows Server 2003 were added in the registry, are no longer supported and therefore are ignored for Windows Server 2008:

* **TcpWindowSize**

HKLM\System\CurrentControlSet\Services\Tcpip\Parameters

* **NumTcbTablePartitions**

HKLM\system\CurrentControlSet\Services\Tcpip\Parameters

* **MaxHashTableSize**

HKLM\system\CurrentControlSet\Services\Tcpip\Parameters

## Network-Related Performance Counters

IPv4

Datagrams received per second.

Datagrams sent per second.

Network Interface > [adapter name]

Bytes received per second.

Bytes sent per second.

Packets received per second.

Packets sent per second.

Output queue length.

This counter is the length of the output packet queue (in packets). If this is longer than 2, delays occur. You should find the bottleneck and eliminate it if you can. Because NDIS queues the requests, this length should always be 0.

Packets received errors.

This counter is the number of incoming packets that contain errors that prevent them from being deliverable to a higher-layer protocol. A zero value does not guarantee that there are no receive errors. The value is polled from the network driver, and it can be inaccurate.

Packets outgoing errors.

Processor

Percent of processor time.

Interrupts per second.

DPCs queued per second.

This counter is an average rate at which DPCs were added to the processor's DPC queue. Each processor has its own DPC queue. This counter measures the rate that DPCs are added to the queue, not the number of DPCs in the queue. It displays the difference between the values that were observed in the last two samples, divided by the duration of the sample interval.

TCPv4

Connection failures.

Segments sent per second.

Segments received per second.

Segments retransmitted per second.

# Performance Tuning for Storage Subsystem

Decisions about how to design or configure storage software and hardware usually consider performance. Performance is always degraded or improved because of trade-offs with other factors such as cost, reliability, availability, power, or ease of use. Trade-offs are made all along the way between application and disk media. File cache management, file system architecture, and volume management translate application calls into individual storage access requests. These requests traverse the storage driver stack and generate streams of commands that are presented to the disk storage subsystem. The sequence and quantity of calls, and the subsequent translation, can improve or degrade performance.

Figure 2 shows the storage architecture, which covers many components in the driver stack. The layered driver model in Windows sacrifices some performance for maintainability and ease of use (in terms of incorporating drivers of varying types into the stack). The following sections discuss tuning guidelines for storage workloads.

**SCSIPORT**

**NTFS**

**VOLMGRX**

**PartMgr**

**FASTFAT**

**VOLMGR**

**Miniport Driver**

**File System Drivers**

**Volume Snapshot and Management Drivers**

**Partition and Class Drivers**

**Port Driver**

**Adapter Interface**

**STORPORT**

**ClassPNP**

**VolSnap**

**DISK**

**ATAPORT**

Figure 2. Storage Driver Stack

## Choosing Storage

The most important considerations in choosing storage systems include the following:

* Understanding the characteristics for current and future storage workloads.
* Understanding that application behavior is essential for both storage subsystem planning and performance analysis.
* Providing necessary storage space, bandwidth, and latency characteristics for current and future needs.
* Selecting a data layout scheme (such as striping), redundancy architecture (such as mirroring), and backup strategy.
* Using a procedure that provides the required performance and data recovery capabilities.
* Using power guidelines, that is, calculating the expected power consumption in total and per-unit volume (such as watts per rack).

When they are compared to 3.5-inch disks, 2.5-inch disks have greatly reduced power consumption but they also are packed more tightly into racks or servers. Note that spinning up disk drives increases power usage, so power-sensitive environments should use arrays that spin up their drives in a staged manner.

The better you understand the workloads on the system, the more accurately you can plan. The following are some important workload characteristics:

* Read:write ratio.
* Sequential/random (temporal and spatial locality).
* Request sizes.
* Interarrival rates, burstiness, and concurrency (patterns of request arrival rates).

### Estimating the Amount of Data to Be Stored

When you estimate how much data to be stored on a new server, consider these issues:

* How much data that is currently stored on servers will be consolidated onto the new server.
* How much replicated data will be stored on the new file server if the server is a file server replica member.
* How much data you must store on the server in the future.

A general guideline is to assume that growth will be faster in the future than it was in the past. Investigate whether your organization plans to hire many employees, whether any groups in your organization plan large projects that will require additional storage, and so on.

You must also consider how much space is used by operating system files, applications, RAID redundancy, log files, and other factors. Table 6 describes some factors that affect server capacity.

Table 6. Factors That Affect Server Capacity

|  |  |
| --- | --- |
| Factor | Required storage capacity |
| Operating system files | At least 1.5 GB.  To provide space for optional components, future service packs, and other items, plan for an additional 3 to 5 GB for the operating system volume. Windows installation can require even more space for temporary files. |
| Paging file | For smaller servers, 1.5 times the amount of RAM, by default.  For servers that have hundreds of gigabytes of memory, the elimination of the paging file is possible; otherwise, the paging file might be limited because of space constraints (available disk capacity). The benefit of a paging file of larger than 50 GB is unclear. |
| Memory dump | Depending on the memory dump file option that you have chosen, as large as the amount of physical memory plus 1 MB.  On servers that have very large amounts of memory, full memory dumps become intractable because of the time that is required to create, transfer, and analyze the dump file. |
| Applications | Varies according to the application. These applications can include antivirus, backup and disk quota software, database applications, and optional components such as Recovery Console, Services for UNIX, and Services for NetWare. |
| Log files | Varies according to the application that creates the log file.  Some applications let you configure a maximum log file size. You must make sure that you have enough free space to store the log files. |
| Data layout and redundancy | Varies.  For more information, see “[Choosing the Raid Level](#_Choosing_the_RAID_Level)” later in this guide. |
| Shadow copies | 10% of the volume, by default But we recommend increasing this size. |

### Choosing a Storage Array Selection

There are many considerations in choosing a storage array and adapters. The choices include the type of storage communication protocols that you use, including the options shown in Table 7.

Table 7. Options for Storage Array Selection

|  |  |
| --- | --- |
| Option | Description |
| Fibre  Channel or SCSI | Fibre Channel enables long glass or copper cables to connect the storage array to the system while it provides high bandwidth. SCSI provides very high bandwidth, but has cable length restrictions. |
| SAS or SATA | These fairly new serial protocols improve performance, reduce cable length limitations, and reduce cost. In the future, SAS and SATA drives will replace much of the SCSI market. |
| Hardware RAID capabilities | For maximum performance and reliability, the storage controllers should offer RAID capabilities. RAID levels 0, 1, 0+1, 5, and 6 are described in Table 7. |
| Maximum storage capacity | Total storage area. |
| Storage bandwidth | The maximum peak and sustained bandwidths at which storage can be accessed is determined by the number of physical disks in the array, the speed of controllers, the type of disk (such as SCSI or Fibre Channel), the hardware RAID, and the adapters that are used to connect the storage array to the system. Of course, the more important values are the achievable bandwidths for the specific workloads to be executed on servers that access the storage. |

### Hardware RAID Levels

Most storage arrays provide some hardware RAID capabilities. Common RAID levels are shown in Table 8.

Table 8. RAID Options

| **Option** | **Description** |
| --- | --- |
| Just a bunch of disks (JBOD) | This is not a RAID level, but instead is the baseline against which to measure RAID performance, cost, and reliability. Individual disks are referenced separately, not as a combined entity.  In some scenarios, JBOD actually provides better performance than striped data layout schemes. For example, when serving multiple lengthy sequential streams, performance is best when a single disk services each stream. Also, workloads that are composed of small, random requests do not improve performance benefits when they are moved from JBOD to a striped data layout.  JBOD is susceptible to static and dynamic “hot spots,” which reduce available storage bandwidth because of load imbalance across the physical drives.  Any physical disk failure results in data loss. However, the loss is limited to the failed drives. In some scenarios, it provides a level of data isolation that can be interpreted as greater reliability. |
| Spanning | This is also not a RAID level, but instead is the simple concatenation of multiple physical disks into a single logical disk. Each disk contains a set of sequential logical blocks. Spanning has the same performance and reliability characteristics as JBOD. |
| RAID 0 (striping) | RAID 0 is a data layout scheme in which sequential logical blocks of a prechosen size (the stripe unit) are laid out in a round-robin manner across multiple disks. It presents a logical disk that stripes disk accesses over a set of physical disks.  For most workloads, a striped data layout provides better performance than JBOD if the stripe unit is appropriately selected based on server workload and storage hardware characteristics. The overall storage load is balanced across all physical drives.  This is the least expensive RAID configuration because all the disk capacity is available for storing the single copy of data.  Because no capacity is allocated for redundant data, RAID 0 does not provide data recovery mechanisms such as those in RAID 1 and RAID 5. Also, the loss of any disk results in data loss on a larger scale than JBOD because the entire file system spread across *n* physical disks is disrupted; every *n*th block of data in the file system is missing. |
| RAID 1 (mirroring) | RAID 1 is a data layout scheme in which each logical block exists on at least two physical disks. It presents a logical disk that consists of a disk mirror pair.  RAID 1 often has worse bandwidth and latency for write operations compared to RAID 0 (or JBOD) This is because data must be written to two or more physical disks. Request latency is based on the slowest of the two (or more) write operations that are necessary to update all copies of the affected data blocks.  RAID 1 can provide faster read operations than RAID 0 because it can read from the least busy physical disk from the mirrored pair.  RAID 1 is the most expensive RAID scheme in terms of physical disks because half (or more) of the disk capacity stores redundant data copies. RAID 1 can survive the loss of any single physical disk. In larger configurations it can survive multiple disk failures, if the failures do not involve all the disks of a specific mirrored disk set.  RAID 1 is the fastest ordinary RAID level for recovery time after a physical disk failure. Only a single disk (the other part of the broken mirror pair) brings up the replacement drive. Note that the second disk is typically still available to service data requests throughout the rebuilding process. |
| RAID 0+1 (striped mirrors) | The combination of striping and mirroring provides the performance benefits of RAID 0 and the redundancy benefits of RAID 1.  This option is also known as RAID 1+0 and RAID 10. |
| RAID 5 (rotated parity) | RAID 5 presents a logical disk composed of multiple physical disks that have data striped across the disks in sequential blocks (stripe units). However, the underlying physical disks have parity information scattered throughout the disk array, as Figure 3 shows.  For read requests, RAID 5 has characteristics that resemble those of RAID 0. However, small RAID 5 writes are much slower than those of JBOD or RAID 0 because each parity block that corresponds to the modified data block requires three additional disk requests. Because four physical disk requests are generated for every logical write, bandwidth is reduced by approximately 75%.  RAID 5 provides data recovery capabilities because data can be reconstructed from the parity. RAID 5 can survive the loss of any one physical disk, as opposed to RAID 1, which can survive the loss of multiple disks as long as an entire mirrored set is not lost.  RAID 5 requires additional time to recover from a lost physical disk compared to RAID 1 because the data and parity from the failed disk can be re‑created only by reading all the other disks in their entirety. Performance during the rebuilding period is severely reduced due only to the rebuilding traffic but also because the reads and writes that target the data that was stored on the failed disk must read all disks (an entire “stripe”) to re-create the missing data.  RAID 5 is less expensive than RAID 1 because it requires only an additional single disk per array, instead of double the total amount of disks in an array.  Power guidelines: RAID 5 has a significant power advantage over mirroring, simply because it uses fewer drives. |
| RAID 6 (double-rotated redundancy) | RAID 6 is basically RAID 5 with additional redundancy built in. Instead of a single block of parity per stripe of data, two blocks of redundancy are included. The second block uses a different redundancy code (instead of parity), which enables data to be reconstructed after the loss of any two disks. Or, disks can be arranged in a two-dimensional matrix, and both vertical and horizontal parity can be maintained.  Power guidelines: RAID 6 has a significant power advantage over mirroring, simply because it uses fewer drives. |

Rotated redundancy schemes (such as RAID 5 and 6) are the most difficult to understand and plan for. Figure 3 shows RAID 5.

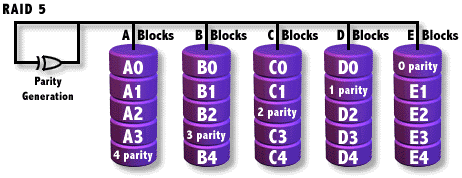


Figure 3. RAID 5 Overview

### Choosing the RAID Level

Each RAID level involves a trade-off between the following factors:

* Cost
* Performance
* Availability
* Reliability
* Power

To determine the best RAID level for your servers, evaluate the read and write loads of all data types and then decide how much you can spend to achieve the performance and availability/reliability that your organization requires. Table 9 describes common RAID levels and their relative costs, performance, availability, and reliability.

Table 9. RAID Trade-Offs

| **Config­uration** | **Performance** | **Reliability** | **Availability** | **Cost, capacity, and power consumed** |
| --- | --- | --- | --- | --- |
| JBOD | **Pros:**   * Concurrent sequential streams to separate disks.   **Cons:**   * Susceptibility to load imbalance. | **Pros:**   * Data isolation; single loss that affects one disk.   **Cons:**   * Data loss after one failure. | **Pros:**   * Single loss that does not prevent access to other disks. | **Pros:**   * Minimum cost. * Minimum power. |
| RAID 0 (striping) | **Pros:**   * Balanced load. * Potential for better response times, throughput, and concurrency.   **Cons:**   * Difficult stripe unit size choice. | **Cons:**   * Data loss after one failure. * Single loss that affects the entire array. | **Cons:**   * Single loss that prevents access to entire array. | **Pros:**   * Minimum cost. * Two-disk minimum. * Minimum power. |
| RAID 1 (mirroring) | **Pros:**   * Two data sources for every read request (up to 100% performance improvement).   **Cons:**   * Writes that must update all mirrors. | **Pros:**   * Single loss and often multiple losses (in large configurations) that are survivable. | **Pros:**   * Single loss and often multiple losses (in large configurations) that do not prevent access. | **Pros:**   * Twice the cost of RAID 0 or JBOD. * Two-disk minimum. * Maximum power. |
| RAID 0+1 (striped mirrors) | **Pros:**   * Two data sources for every read request (up to 100% performance improvement). * Balanced load. * Potential for better response times, throughput, and concurrency.   **Cons:**   * Writes that must update mirrors. * Difficult stripe unit size choice. | **Pros:**   * Single loss and often multiple losses (in large configurations) that are survivable. | **Pros:**   * Single loss and often multiple losses (in large configurations) that do not prevent access. | **Pros:**   * Twice the cost of RAID 0 or JBOD. * Four-disk minimum. * Maximum power. |
| RAID 5 (rotated parity) | **Pros:**   * Balanced load. * Potential for better read response times, throughput, and concurrency.   **Cons:**   * Up to 75% write performance reduction because of RMW. * Decreased read performance in failure mode. * All sectors that must be read for reconstruction; major slowdown. * Danger of data in invalid state after power loss and recovery. | **Pros:**   * Single loss survivable; “in-flight” write requests might still become corrupted.   **Cons:**   * Multiple losses that affect entire array. * After a single loss, array that is vulnerable until reconstructed. | **Pros:**   * Single loss does not prevent access.   **Cons:**   * Multiple losses that prevent access to entire array. * To speed reconstruction, application access that might be slowed or stopped. | **Pros:**   * One additional disk required. * Three-disk minimum. * Only one more disk to power. |
| RAID 6 (two separate erasure codes) | **Pros:**   * Balanced load. * Potential for better read response times, throughput, and concurrency.   **Cons:**   * Up to 83% write performance reduction because of multiple RMW. * Decreased read performance in failure mode. * All sectors that must be read for reconstruction: major slowdown. * Danger of data in invalid state after power loss and recovery. | **Pros:**   * Single loss survivable; “in‑flight” write requests might still be corrupted.   **Cons:**   * >2 losses that affect entire array. * After 2 losses, an array that is vulnerable until reconstructed. | **Pros:**   * Single loss that does not prevent access.   **Cons:**   * >2 losses that prevent access to entire array. * To speed reconstruction, application access that might be slowed or stopped. | **Pros:**   * Two additional disks required. * Five-disk minimum. * Only two more disks to power. |

The following are sample uses for various RAID levels:

* JBOD: Concurrent video streaming.
* RAID 0: Temporary or reconstructable data, workloads that can develop hot spots in the data, and workloads with high degrees of unrelated concurrency.
* RAID 1: Database logs, and critical data and concurrent sequential streams.
* RAID 0+1: A general-purpose combination of performance and reliability for critical data, workloads with hot spots, and high-concurrency workloads.
* RAID 5: Web pages, semicritical data, workloads without small writes, scenarios in which capital and operating costs are an overriding factor, and read-dominated workloads.
* RAID 6: Data mining, critical data (assuming quick replacement or hot spares), workloads without small writes, scenarios in which cost or power is a major factor, and read-dominated workloads.

If you use more than two disks, RAID 0+1 is usually a better solution than RAID 1.

To determine the number of physical disks that you should include in RAID 0, RAID 5, and RAID 0+1 virtual disks, consider the following information:

* Bandwidth (and often response time) improves as you add disks.
* Reliability, in terms of mean time to failure for the array, decreases as you add disks.
* Usable storage capacity increases as you add disks, but so does cost.
* For striped arrays, the trade-off is in data isolation (small arrays) and better load balancing (large arrays). For RAID 1 arrays, the trade-off is in better cost/capacity (mirrors—that is, a depth of two) and the ability to withstand multiple disk failures (shadows—that is, depths of three or even four). Read and write performance issues can also affect RAID 1 array size. For RAID 5 arrays, the trade-off is better data isolation and mean time between failures (MTBF) for small arrays and better cost/capacity/power for large arrays.
* Because hard drive failures are not independent, array sizes must be limited when the array is made up of actual physical disks (that is, a bottom-tier array). The exact amount of this limit is very difficult to determine.

The following is the array size guideline with no available hardware reliability data:

* Bottom-tier RAID 5 arrays should not extend beyond a single desk-side storage tower or a single row in a rack-mount configuration. This means approximately 8 to 14 physical disks for modern 3.5-inch storage enclosures. Smaller 2.5-inch disks can be racked more densely and therefore may require dividing into multiple arrays per enclosure.
* Bottom-tier mirrored arrays should not extend beyond two towers or rack-mount rows, with data being mirrored between towers or rows when possible. These guidelines help avoid or reduce the decrease in MTBF that is caused by using multiple buses, power supplies, and so on from separate storage enclosures.

### Selecting a Stripe Unit Size

The Windows volume manager stripe unit is fixed at 64 KB. Hardware solutions can range from 4 KB to 1 MB and even more. Ideal stripe unit size maximizes the disk activity without unnecessarily breaking up requests by requiring multiple disks to service a single request. For example, consider the following:

* One long stream of sequential requests on JBOD uses only one disk at a time. To keep all disks in use for such a workload, the stripe unit should be at least 1/*n* where *n* is the request size.
* For *n* streams of small serialized random requests, if *n* is significantly greater than the number of disks and if there are no hot spots, striping does not increase performance over JBOD. However, if hot spots exist, the stripe unit size must maximize the possibility that a request will not be split while it minimizes the possibility of a hot spot falling entirely within one or two stripe units. You might choose a low multiple of the typical request size, such as 5X or 10X, especially if the requests are on some boundary (for example, 4 KB or 8 KB).
* If requests are large and the average (or perhaps peak) number of outstanding requests is smaller than the number of disks, you might need to split some so that all disks are being used. Interpolate from the previous two examples. For example, if you have 10 disks and 5 streams of requests, split each request in half. (Use a stripe unit size equal to half the request size.)
* Optimal stripe unit size increases with concurrency, burstiness, and typical request sizes.
* Optimal stripe unit size decreases with sequentiality and with good alignment between data boundaries and stripe unit boundaries.

### Determining the Volume Layout

Placing individual workloads into separate volumes has advantages. For example, you can use one volume for the operating system or paging space and one or more volumes for shared user data, applications, and log files. The benefits include fault isolation, easier capacity planning, and easier performance analysis.

You can place different types of workloads into separate volumes on different virtual disks. Using separate virtual disks is especially important for any workload that creates heavy sequential loads such as log files, where a single set of disks (that compose the virtual disk) can be dedicated to handling the disk I/O that the updates to the log files create. Placing the paging file on a separate virtual disk might provide some improvements in performance during periods of high paging.

There is also an advantage to combining workloads on the same physical disks, if the disks do not experience high activity over the same time period. This is basically the partnering of hot data with cold data on the same physical drives.

The “first” partition on a volume usually uses the outermost tracks of the underlying disks and therefore provides better performance.

## Storage-Related Parameters

On Windows Server 2008, you can adjust the following registry parameter for high-throughput scenarios.

### NumberOfRequests

This driver/device-specific parameter is passed to a miniport when it is initialized. A higher value might improve performance and enable Windows to give more disk requests to a logical disk, which is most useful for hardware RAID adapters that have concurrency capabilities. This value is typically set by the driver when it is installed, but you can set it manually through the following registry entry:

HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services

\MINIPORT\_ADAPTER\Parameters\DeviceN\NumberOfRequests (REG\_DWORD)

Replace miniport\_adapter with the specific adapter name. Make an entry for each device, and in each entry replace *DeviceN* with Device1, Device2, and so on, depending on the number of devices that you are adding. For this setting to take effect, a reboot is sometimes required. But for Storport miniports, only the adapters must be “rebooted” (that is, disabled and re-enabled). For example, for two Emulex miniport adapters whose miniport driver name is lp6nds35, you would create the following registry entries set to 96:

HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services\lp6nds35\Parameters  
\Device0\NumberOfRequests

HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services\lp6nds35\Parameters  
\Device1\NumberOfRequests

The following parameters do not apply to Windows Server 2008:

CountOperations

HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Session Manager\I/O System\

DontVerifyRandomDrivers

HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Session Manager  
\Memory Management\

### I/O Priorities

Windows Server 2008 can specify an internal priority level on individual I/Os. Windows primarily uses this ability to de-prioritize background I/O activity and to give precedence to response-sensitive I/Os (such as, multimedia). However, extensions to file system APIs let applications specify /IO priorities per handle. The storage stack code to sort out and manage I/O priorities has overhead, so if some disks will be targeted only by a single priority of I/Os (such as a SQL database disk), you can improve performance by disabling the I/O priority management for those disks by setting the following registry entry to zero:

HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Control\DeviceClasses  
\{Device GUID}\DeviceParameters\Classpnp\IdlePrioritySupported

## Storage-Related Performance Counters

### Logical Disk and Physical Disk

On servers that have heavy I/O workloads, the disk counters should be enabled on a sampling basis or specifically to diagnose storage-related performance issues to avoid incurring up to a 1-percent overhead penalty from having the counters running.

The same counters are valuable in both the logical and physical disk counter objects. Logical disk statistics are tracked by the volume manager (or managers), and physical disk statistics are tracked by the partition manager.

Note: If the Windows standard stacked driver’s scheme is circumvented for some controller, so-called “monolithic” drivers can assume the role of partition manager or volume manager. If so, the monolithic driver writer must supply the same counters through the Windows Management Instrumentation (WMI) interface:

* % Disk Read Time, % Disk Time, % Disk Write Time, % Idle Time

These counters are of little value when multiple physical drives are behind logical disks. Imagine a subsystem of 100 physical drives presented to the operating system as five disks, each backed by a 20-disk RAID 0+1 array. Now imagine that the administrator spans the five physical disks that have one logical disk, volume *x*. One can assume that any serious system that needs that many physical disks has at least one outstanding request to *x* at the same time. This makes the volume appear to be 100% busy and 0% idle, when in fact the 100-disk array could be up to 99% idle.

* Average Disk Bytes / { Read | Write | Transfer }

This counter collects average, minimum, and maximum request sizes. If possible, individual or sub-workloads should be observed separately. Multimodal distributions cannot be differentiated by using average values if the request types are consistently interspersed.

* Average Disk Queue Length, Average Disk { Read | Write } Queue Length

These counters collect concurrency data, including burstiness and peak loads. Guidelines for queue lengths are given later in this guide. These counters represent the number of requests in flight below the driver that takes the statistics. This means that the requests are not necessarily queued but could actually be in service or completed and on the way back up the path. Possible in-flight locations include the following:

Waiting in an ATAport, SCSIPort, or Storport queue.

Waiting in a queue in a miniport driver.

Waiting in a disk controller queue.

Waiting in an array controller queue.

Waiting in a hard disk queue (that is, on board a physical disk).

Actively receiving service from a physical disk.

Completed, but not yet back up the stack to where the statistics are collected.

* Average Disk second / {Read | Write | Transfer}

These counters collect disk request response time data and possibly extrapolate service time data. They are probably the most straightforward indicators of storage subsystem bottlenecks. Guidelines for response times are given later in this guide. If possible, individual or sub-workloads should be observed separately. Multimodal distributions cannot be differentiated by using Perfmon if the requests are consistently interspersed.

* Current Disk Queue Length

This counter instantly measures the number of requests in flight and therefore is subject to extreme variance. Therefore, this counter is not useful except to check for the existence of many short bursts of activity.

* Disk Bytes / second, Disk {Read | Write } Bytes / second

This counter collects throughput data. If the sample time is long enough, a histogram of the array’s response to specific loads (queues, request sizes, and so on) can be analyzed. If possible, individual or sub-workloads should be observed separately.

* Disk {Reads | Writes | Transfers } / second

This counter collects throughput data. If the sample time is long enough, a histogram of the array’s response to specific loads (queues, request sizes, and so on) can be analyzed. If possible, individual or sub-workloads should be observed separately.

* Split I/O / second

This counter is useful only if the value is not in the noise. If it becomes significant, in terms of split I/Os per second per physical disk, further investigation could be needed to determine the size of the original requests that are being split and the workload that is generating them.

### Processor

* % DPC Time, % Interrupt Time, % Privileged Time

If interrupt time and DPC time are a large part of privileged time, the kernel is spending a long time processing I/Os. Sometimes, it is best to keep interrupts and DPCs affinitized to only a few CPUs on a multiprocessor system, to improve cache locality. Other times, it is best to distribute the interrupts and DPCs among many CPUs to prevent the interrupt and DPC activity from becoming a bottleneck.

* DPCs Queued / second

This counter is another measurement of how DPCs are using CPU time and kernel resources.

* Interrupts / second

This counter is another measurement of how interrupts are using CPU time and kernel resources. Modern disk controllers often combine or coalesce interrupts so that a single interrupt causes the processing of multiple I/O completions. Of course, it is a trade-off between delaying interrupts (and therefore completions) and amortizing CPU processing time.

### Power Protection and Advanced Performance Option

There are two performance-related options for every disk under **Disk > Properties > Policies**. Enabling write caching lets writes be completed immediately after they have been cached in the storage subsystem. Note that with this action a period of time passes during which a power failure or other catastrophic event could result in data loss. However, this period is typically fairly short because write caches in the storage subsystem are usually flushed during any period of idle activity. Alternately, you can use time-outs at the cache level to force dirty data out of the cache even if other active requests exist.

The advanced performance option strips all write-through flags from disk requests and removes all flush-cache commands. The assumption is that if you have power protection on your I/O path you do not need to worry about those two pieces of functionality. By definition, any written data is safe and “in-order” after it is copied into power-protected storage subsystem hardware, just as if it had been written to the physical disk media.

### Block Alignment (DISKPART)

NTFS aligns its metadata and data clusters to partition boundary by increments of the cluster size (which was selected during file system creation or defaulted to 4 KB). In earlier releases of Windows, the partition boundary offset for a specific disk partition could be misaligned, when it was compared to array disk stripe unit boundaries. This caused requests to be unintentionally split across multiple disks. To force alignment, you must use diskpar.exe or diskpart.exe at the time that the partition is created.

In Windows Server 2008, partitions are automatically offset by 1 MB, which provides good alignment for the power-of-two stripe unit sizes that are typically found in hardware. If the stripe unit size is set to a size that is greater than 1 MB, the alignment issue is much less of a problem because small requests rarely cross large stripe unit boundaries. Note that Windows Server 2008 defaults to a smaller power-of-two offset for small drives.

If alignment is still a problem even with the default offset, you can use diskpart.exe to force alternative alignments at the time of partition creation.

### Solid-State and Hybrid Drives

Previously, the cost of large quantities of nonvolatile memory was prohibitive for server configurations. Exceptions include aerospace or military applications in which the shock and vibration sensitivity of flash memory is highly desirable. Newer laptops and desktop systems have begun to incorporate flash memory in the form of “hybrid” disk drives. In these configurations, Windows can explicitly request that some data blocks be cached in a drive’s nonvolatile memory and other blocks be sent directly to the magnetic media. Because the amount of flash memory is quite small when it is compared to the amount of data that can be stored on the platters, the cost is acceptable. This is especially true when one considers the other benefits of flash memory: improved power and greater tolerance of shock, vibration, and heat.

As the cost of flash memory continues to decrease, it becomes more possible to improve storage subsystem response time on servers. The typical vehicle for incorporating nonvolatile memory in a server is the solid-state disk (SSD). The most cost-effective way is to place only the “hottest” data of a workload onto nonvolatile memory. In Windows Server 2008, partitioning can be performed only by applications that store data on the SSD. Windows Server 2008 does not try to dynamically determine what data should optimally be stored on SSDs.

### Response Times

You can use tools such as Perfmon to obtain data on disk request response times. Write requests that enter a writeback hardware cache often have very low response times (less than 1 ms) because completion depends on dynamic RAM (DRAM) instead of disk speeds. The data is written back to disk media in the background. As the workload begins to saturate the cache, response times increase until the write cache’s only benefit is potentially a better ordering of requests to reduce positioning delays.

For JBOD arrays, reads and writes have approximately the same performance characteristics. With modern hard disks, positioning delays for random requests are 5 to 15 ms. Smaller 2.5-inch drives have shorter positioning distances and lighter actuators, so they generally provide faster seek times than comparable larger 3.5-inch drives. Positioning delays for sequential requests should be insignificant except for write-through streams, where each positioning delay should approximate the required time for a complete disk rotation.

Transfer times are usually less significant when they are compared to positioning delays, except for sequential requests and large requests (larger than 256 KB) that are instead dominated by disk media access speeds as the requests become larger or more sequential. Modern hard disks access their media at 25 to 125 MB per second depending on rotation speed and sectors per track, which varies across a range of blocks on a specific disk model. Outermost tracks can have up to twice the sequential throughput of innermost tracks.

If the stripe unit size of a striped array is well chosen, each request is serviced by a single disk—except for a low-concurrency workload. So, the same general positioning and transfer times still apply.

For mirrored arrays, a write completion might be required to wait for both disks to complete the request. Depending on how the requests are scheduled, the two completions of the requests could take a long time. However, although writes generally should not take twice the time to complete for mirrored arrays, they are probably slower than JBOD. Or, reads can experience a performance increase if the array controller is dynamically load-balancing or considering spatial locality.

For RAID 5 arrays (rotated parity), small writes become four separate requests in the typical read-modify-write scenario. In the best case, this is approximately the equivalent of two mirrored reads plus a full rotation of the disks, if you assume that the Read/Write pairs continue in parallel. RAID 6 incurs an even greater performance hit for writes because each RAID 6 small write request becomes three reads plus three writes.

You must consider the performance affect of redundant arrays on read and write requests when you plan subsystems or analyze performance data. For example, Perfmon might show that 50 writes per second are being processed by volume x, but in reality this could mean 100 requests per second for a mirrored array, 200 requests per second for a RAID 5 array, or even more than 200 requests per second if the requests are split across stripe units.

The following are response time guidelines if no workload details are available. For a lightly loaded system, average write response times should be less than 25 ms on RAID 5 and less than 15 ms on non-RAID 5 disks. Average read response times should be less than 15 ms. For a heavily loaded system that is not saturated, average write response times should be less than 75 ms on RAID 5 and less than 50 ms on non-RAID 5 disks. Average read response times should be less than 50 ms.

### Queue Lengths

Several opinions exist about what constitutes excessive disk request queuing. This guide assumes that the boundary between a busy disk subsystem and a saturated one is a persistent average of two requests per physical disk. A disk subsystem is near saturation when every physical disk is servicing a request and has at least one queued-up request to maintain maximum concurrency—that is, to keep the data pipeline flowing. Note that in this guideline, disk requests split into multiple requests (because of striping or redundancy maintenance) are considered multiple requests.

This rule has caveats, because most administrators do not want all physical disks constantly busy. But because disk workloads are generally bursty, this rule is more likely applied over shorter periods of (peak) time. Requests are typically not uniformly spread among all hard disks at the same time, so the administrator must consider deviations between queues—especially for bursty workloads. Conversely, a longer queue provides more opportunity for disk request schedulers to reduce positioning delays or optimize for full-stripe RAID 5 writes or mirrored read selection.

Because hardware has an increased capability to queue up requests—either through multiple queuing agents along the path or merely agents with more queuing capability—increasing the multiplier threshold might allow more concurrency within the hardware. This creates a potential increase in response time variance, however. Ideally, the additional queuing time is balanced by increased concurrency and reduced mechanical positioning times.

The following is a queue length target to use when few workload details are available. For a lightly loaded system, the average queue length should be less than one per physical disk, with occasional spikes of 10 or less. If the workload is write heavy, the average queue length above a mirrored controller should be less than 0.6 per physical disk and less than 0.3 per physical disk above a RAID 5 controller. For a heavily loaded system that is not saturated, the average queue length should be less than 2.5 per physical disk, with infrequent spikes up to 20. If the workload is write heavy, the average queue length above a mirrored controller should be less than 1.5 per physical disk and less than 1.0 per physical disk above a RAID 5 controller. For workloads of sequential requests, larger queue lengths can be tolerated because services times and therefore response times are much shorter than those for a random workload.

For more details on Windows storage performance, see “[Disk Subsystem Performance Analysis for Windows](http://www.microsoft.com/whdc/device/storage/subsys_perf.mspx).”

# Performance Tuning for Web Servers

## Selecting the Proper Hardware for Performance

It is important to select the proper hardware to satisfy the expected Web load (remembering average load, peak load, capacity, growth plans, and response times). Hardware bottlenecks limit the effectiveness of software tuning. “[Performance Tuning for Server Hardware](#_Performance_Tuning_for_Networking)” earlier in this guide provides recommendations for hardware to avoid the following performance constraints:

* Slow CPUs offer limited processing power for ASP, ASP.NET, and SSL scenarios.
* A small L2 processor cache might adversely affect performance.
* A limited amount of memory affects the number of sites that can be hosted, how many dynamic content scripts (such as ASP.NET) that can be stored, and the number of application pools or worker processes.
* Networking becomes a bottleneck because of an inefficient networking adapter.
* The file system becomes a bottleneck because of an inefficient disk subsystem or storage adapter.

## Operating System Practices

If possible, do a clean installation of the operating system software. Upgrading could leave outdated, unwanted, or suboptimal registry settings and previously installed services and applications that consume resources if they are started automatically. If another operating system is installed and must be kept, you should install the new operating system on a different partition. Otherwise, the new installation overwrites the settings under Program Files\Common Files.

To reduce disk access interference, keep the system pagefile, operating system, Web data, ASP template cache, and Internet Information Services (IIS) log on separate physical disks if possible.

To reduce the contention of system resources, install SQL and IIS on different servers if possible.

Avoid installing nonessential services and applications. In some cases, it might be worthwhile to disable services that are not required on a system.

## Tuning IIS 7.0

IIS 7.0 uses a process model similar to that of IIS 6.0. A kernel-mode HTTP listener (Http.sys) receives and routes HTTP requests (and can even satisfy requests from its response cache). Worker processes register for URL subspaces, and Http.sys routes the request to the appropriate process (or set of processes for application pools).

Figure 4 shows the difference between the IIS 6.0 and IIS 7.0 process models. IIS 6.0 kept a single copy of the metabase in a global process, inetinfo.exe. IIS 7.0 no longer uses the metabase and instead loads XML configuration files that are located alongside Web content. Each worker process loads a unique copy of configuration. IIS 7.0 also implements an “integrated pipeline.” The integrated pipeline model exposes extensibility.



Figure 4. Process Models for IIS 6.0 and IIS 7.0

The IIS 7.0 process relies on the kernel-mode Web driver, Http.sys. Http.sys is responsible for connection management and request handling. The request can be either served from the Http.sys cache or handed to a worker process for further handling (see ). Multiple worker processes can be configured, which provides isolation at a reduced cost.

Http.sys includes a response cache. When a request matches an entry in the response cache, Http.sys sends the cache response directly from kernel mode. shows the request flow from the network through Http.sys (and possibly up to a worker process). Some Web application platforms, such as ASP.NET, provide mechanisms to enable any dynamic content to be cached in the kernel cache. The static file handler in IIS 7.0 automatically caches frequently requested files in http.sys.



Figure 5. Request Handling in IIS 7.0

Because a Web server has a kernel-mode and a user-mode component, both components must be tuned for optimal performance. Therefore, tuning IIS 7.0 for a specific workload includes configuring the following:

* Http.sys (the kernel-mode driver) and the associated kernel-mode cache.
* Worker processes and user-mode IIS, including application pool configuration.
* Certain tuning parameters that affect performance, which are discussed in the following sections.

## Kernel-Mode Tunings

Performance-related Http.sys settings fall into two broad categories: cache management, and connection and request management. All registry settings are stored under the following entry:

HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services\Http\Parameters

If the HTTP service is already running, it must be stopped and restarted for the changes to take effect.

### Cache Management Settings

One benefit that Http.sys provides is a kernel-mode cache. If the response is in the kernel cache, you can satisfy an HTTP request entirely from kernel mode, which significantly lowers the CPU cost of handling the request. However, the kernel-mode cache of IIS 7.0 is a physical memory-based cache and the cost of an entry is the memory that it occupies.

An entry in the cache is helpful only when it is used. However, the entry always uses physical memory, whether the entry is being used or not. You must evaluate the usefulness of an item in the cache (the difference made in being able to serve it from the cache makes) and its cost (the physical memory occupied) over the lifetime of the entry by considering the available resources (CPU and physical memory) and the workload requirements. Http.sys tries to keep only useful, actively accessed items in the cache, but you can increase the performance of the Web server by tuning the Http.sys cache for particular workloads.

The following are some useful settings for the Http.sys kernel-mode cache:

* **UriEnableCache.**Default value 1.

A nonzero value enables the kernel-mode response and fragment cache. For most workloads, the cache should remain enabled. Consider disabling the cache if you expect very low response and fragment cache usage.

* **UriMaxCacheMegabyteCount.** Default value 0.

A nonzero value specifies the maximum memory that is available to the kernel cache. The default value, 0, enables the system to automatically adjust how much memory is available to the cache. Note that specifying the size sets only the maximum and the system might not let the cache grow to the specified size.

* **UriMaxUriBytes.** Default value 262144 bytes (256 KB).

This is the maximum size of an entry in the kernel cache. Responses or fragments larger than this are not cached. If you have enough memory, consider increasing the limit. If memory is limited and large entries are crowding out smaller ones, it might be helpful to lower the limit.

* **UriScavengerPeriod.** Default value 120 seconds.

The Http.sys cache is periodically scanned by a scavenger, and entries that are not accessed between scavenger scans are removed. Setting the scavenger period to a high value reduces the number of scavenger scans. However, the cache memory usage might increase because older, less frequently accessed entries can remain in the cache. Setting the period to too low a value causes more frequent scavenger scans and might result in too many flushes and cache churn.

### Request and Connection Management Settings

Http.sys also manages incoming HTTP/HTTPS connections and is the first layer to handle requests on those connections. It uses internal data structures to keep information about connections and requests. Although such data structures can be created and freed on demand, it is more CPU-efficient to reserve some in look-aside lists. Keeping such reserves helps Http.sys handle fluctuations in load with less CPU usage. Note that load fluctuations are not necessarily the result of fluctuations in externally applied load. Internal optimizations to promote batch processing, and even interrupt moderation, can result in load fluctuations and spikes.

The reserves help reduce CPU usage and latency, and they increase Web server capacity but increase memory usage. When you tune the request and connection management behavior of Http.sys, you should remember the resources that are available to the server, your performance goals, and the characteristics of the workload. Use the following request and connection management settings:

* **MaxConnections**

This value controls the number of concurrent connections that Http.sys allows. Each connection consumes nonpaged pool, a precious and limited resource. The default is determined very conservatively to limit how much nonpaged pool is used for connections. On a dedicated Web server that has ample memory, you should set the value higher if you expect a significant concurrent connection load. A high value can result in increased nonpaged pool usage, so make sure to use a value that is appropriate for the system.

* **IdleConnectionsHighMark**, **IdleConnectionsLowMark**, and **IdleListTrimmerPeriod**

These values control the handling of connection structures that are currently not being used: how many must be available at any time (to handle spikes in connection load), the low and high watermarks for the free list, and the frequency of connection structure trimming and replenishment.

* **RequestBufferLookasideDepth** and **InternalRequestLookasideDepth**

These values control the handling of data structures that are related to buffer management and how many are kept in reserve to handle load fluctuations.

## User-Mode Settings

The settings in this section affect the IIS 7.0 worker process behavior. Most of these settings can be found in the %SystemRoot%\system32\inetsrv\config  
\applicationHost.config XML configuration file. Use either appcmd.exe or the IIS 7.0 management console to change them. Most settings are automatically detected and do not require a restart of the IIS 7.0 worker processes or Web Application Server.

### User-Mode Cache Behavior Settings

This section describes the settings that affect caching behavior in IIS 7.0. The user-mode cache is implemented as a module that listens to the global caching events that the integrated pipeline fires. To completely disable the user-mode cache, remove the FileCacheModule (cachfile.dll) module from the list of installed modules in the *system.webServer/globalModules* configuration section in applicationHost.config.

system.webServer/caching

| **Attribute** | **Description** | **Default** |
| --- | --- | --- |
| *enabled* | Disables the user-mode IIS cache when set to false. When the cache hit rate is very small, you can disable the cache completely to avoid the overhead that is associated with the cache code path. Disabling the user mode cache does not disable the kernel-mode cache. | True |
| *enableKernelCache* | Disables the kernel-mode cache when set to false. | True |
| *maxCacheSize* | Limits the IIS user-mode cache size to the specified size in megabytes. IIS adjusts the default depending on available memory. Choose the value carefully based on the size of the hot set (the set of frequently accessed files) versus the amount of RAM or the IIS process address space, which is limited to 2 GB on 32-bit systems. | 0 |
| *maxResponseSize* | Lets files up to the specified size be cached. The actual value depends on the number and size of the largest files in the dataset versus the available RAM. Caching large, frequently requested files can reduce CPU usage, disk access, and associated latencies. The default value is 256 KB. | 262144 |

### Compression Behavior Settings

IIS 7.0 compresses static content by default. Compression reduces bandwidth usage but increases CPU usage. Compressed content is cached in the kernel-mode cache if possible. IIS 7.0 lets compression be controlled independently for static and dynamic content. Static content typically refers to content that does not change, such as GIF or HTM files. Dynamic content is typically generated by scripts or code on the server, that is, ASP.NET pages. You can customize the classification of any particular extension as static or dynamic.

To completely disable compression, remove *StaticCompressionModule* and *DynamicCompressionModule* from the list of modules in system.webServer/globalModules.

system.webServer/httpCompression

| **Attribute** | **Description** |
| --- | --- |
| *staticCompressionEnableCpuUsage,*  *staticCompressionDisableCpuUsage,*  *dynamicCompressionEnableCpuUsage,*  *dynamicCompressionDisableCpuUsage* | Enables or disables compression if the current percentage CPU usage goes above or below specified limits.  IIS 7.0 automatically disables compression if steady-state CPU increases above the disable threshold. Compression is re-enabled if CPU drops below the enable threshold.  *The default values are 100, 50, 90, and 50, respectively.* |
| *directory* | Specifies the directory in which compressed versions of static files are temporarily stored and cached. Consider moving this directory off the system drive if it is accessed frequently.  *The default value is %SystemDrive%\inetpub\temp \IIS Temporary Compressed Files.* |
| *doDiskSpaceLimiting* | Specifies whether a limit exists for how much disk space all compressed files, which are stored in the compression directory that is specified by directory, can occupy.  *The default value is “true.”* |
| *maxDiskSpaceUsage* | Specifies the number of bytes of disk space that compressed files can occupy in the compression directory.  This setting might need to be increased if the total size of all compressed content is too large.  *The default value is 100 MB.* |

system.webServer/urlCompression

| **Attribute** | **Description** | **Default** |
| --- | --- | --- |
| *doStaticCompression* | Specifies whether static content is compressed. | True |
| *doDynamicCompression* | Specifies whether dynamic content is compressed. | False |

**Note:** For IIS 7.0 servers that have low average CPU usage, consider enabling compression for dynamic content, especially if responses are large. This should first be done in a test environment to assess the effect on the CPU usage from the baseline.

### Tuning the Default Document List

The default document module handles HTTP requests for the root of a directory and translates them into requests for a specific file, such as default.htm or index.htm. On average, around 25 percent of all requests on the Internet go through the default document path. This varies significantly for individual sites. When an HTTP request does not specify a file name, the default document module linearly walks the list of allowed default documents, searching for each one in the file system. This can adversely affect performance, especially if reaching the content requires making a network roundtrip or touching a disk.

You can avoid the overhead by selectively disabling default documents and by reducing or ordering the list of documents. For Web sites that use a default document, you should reduce the list to only the default document types that are used. Additionally, order the list so that it begins with the most frequently accessed default document file name. Finally, you can selectively set the default document behavior on particular URLs by using custom configuration inside a location tag in applicationHost.config or by inserting a web.config file directly in the content directory. This allows a hybrid approach, which enables default documents only where they are necessary and setting the list to the correct file name for each URL.

To disable default documents completely, remove DefaultDocumentModule from the list of modules in the system.webServer/globalModules section in applicationHost.config.

system.webServer/defaultDocument

|  |  |  |
| --- | --- | --- |
| Attribute | Description | Default |
| *enabled* | Specifies that default documents are enabled. | True |
| *<files> element* | Specifies the file names that are configured as default documents.  *The default list is Default.htm, Default.asp, index.htm, index.html, iisstart.htm, and default.aspx.* | Not applicable |

### Central Binary Logging

Binary IIS logging reduces CPU usage, disk I/O, and disk space usage. Central binary logging is directed to a single file in binary format, regardless of the number of hosted sites. Parsing binary-format logs requires a post-processing tool.

You can enable central binary logging by setting the *centralLogFileMode* attribute to CentralBinary and setting the *enabled* attribute to “true.” Consider moving the location of the central log file off the system partition and onto a dedicated logging partition to avoid contention between system activities and logging activities.

system.applicationHost/log

| **Attribute** | **Description** | **Default** |
| --- | --- | --- |
| *centralLogFileMode* | Specifies the logging mode for a server. Change this value to CentralBinary to enable central binary logging. | Site |

system.applicationHost/log/centralBinaryLogFile

| **Attribute** | **Description** | **Default** |
| --- | --- | --- |
| *enabled* | Specifies whether central binary logging is enabled. | False |
| *directory* | Specifies the directory where log entries are written.  *The default directory is %SystemDrive%\inetpub\logs\LogFiles.* | See des-cription |

### Application and Site Tunings

The following settings relate to application pool and site tunings.

system.applicationHost/applicationPools/applicationPoolDefaults

| **Attribute** | **Description** | **Default** |
| --- | --- | --- |
| *queueLength* | Indicates to the Universal Listener how many requests are made to queue for an application pool before future requests are rejected. When the set value for this property is exceeded, IIS rejects subsequent requests with a 503 error.  Consider increasing this for applications that communicate with high-latency back-end data stores if 503 errors are observed. | 1000 |
| *enable32BitAppOnWin64* | When true, enables a 32-bit application to run on a computer that has a 64-bit processor.  Consider enabling 32-bit mode if memory consumption is a concern. Because pointer sizes and instruction sizes are smaller, 32-bit applications use less memory than 64-bit applications. The drawback to running 32-bit applications on a 64-bit machine is that user-mode address space is limited to 4 GB. | False |

system.applicationHost/sites/VirtualDirectoryDefault

| **Attribute** | **Description** | **Default** |
| --- | --- | --- |
| *enabled* | Specifies whether IIS looks for Web.config files in content directories lower than the current level (true) or does not look for Web.config files in content directories lower than the current level (false).  When configuration is queried in the IIS 7.0 pipeline, it is not known whether a URL (/<name>.htm) is a reference to a directory or a file name. By default, IIS 7.0 must assume that /<name>.htm is a reference to a directory and search for configuration in a /<name>.htm/web.config file. This results in an additional file system operation that can be costly.  By imposing a simple limitation, which allows configuration only in virtual directories, IIS 7.0 can then know that unless /<name>.htm is a virtual directory it should not look for a configuration file. Skipping the additional file operations can significantly improve performance to Web sites that have a very large set of randomly accessed static content. | True |

### Managing IIS 7.0 Modules

IIS 7.0 has been refactored into multiple, user-pluggable modules to support a more modular structure. This refactorization has a small cost. For each module present, the integrated pipeline must call into the module for every event that is relevant to the module. This happens regardless of whether the module must do any work. You can conserve CPU cycles and memory by removing all modules that do not have relevance to a particular Web site.

A Web server that is tuned only for simple static files might include only the following five modules: *UriCacheModule*, *HttpCacheModule*, *StaticFileModule*, *AnonymousAuthenticationModule,* and *HttpLoggingModule*.

To remove modules from applicationHost.config, remove all references to the module from the *system.webServer/handlers* and *system.webServer/modules* sectionsin addition to the module declaration in *system.webServer/globalModules*.

### Classic ASP Settings

The following settings apply only to classic ASP pages and do not affect ASP.NET settings. For performance recommendations on ASP.NET, see the MSDN® article “[10 Tips for Writing High-Performance Web Applications](http://go.microsoft.com/fwlink/?LinkId=98290).”

system.webServer/asp/cache

| **Attribute** | **Description** | **Default** |
| --- | --- | --- |
| *diskTemplateCacheDirectory* | If possible, set to a platter not in heavy use, for example, not shared with the operating system, pagefile, IIS log, or other frequently accessed content.  *The default directory is %SystemDrive%\inetpub\temp \ASP Compiled Templates.* | See des­cription |
| *maxDiskTemplateCacheFiles* | This specifies whether disk caching of ASP script templates is enabled. Compiling the ASP templates is a processor-intensive task. Memory constraints limit the number of templates that can be cached in memory. Fetching compiled templates from the disk template cache incurs less cost than compiling templates that do not fit into the ASP memory cache. | True |
| *scriptFileCacheSize* | Set to as many ASP templates as memory limits allow. | 250 |
| *scriptEngineCacheMax* | Set to as many script engines as memory limits allow. | 125 |

system.webServer/asp/limits

| **Attribute** | **Description** | **Default** |
| --- | --- | --- |
| *processorThreadMax* | Specifies the maximum number of worker threads per processor that ASP can create. Increase if the current setting is insufficient to handle the load, possibly causing errors when it is serving some requests or under-usage of CPU resources. | 25 |

system.webServer/asp/comPlus

| **Attribute** | **Description** | **Default** |
| --- | --- | --- |
| *executeInMta* | Set to “true” if errors or failures are detected while it is serving some ASP content. This can occur, for example, when hosting multiple isolated sites in which each site runs under its own worker process. Errors are typically reported from COM+ in the event viewer. This setting enables the multithreaded apartment model in ASP. | False |

### ASP.NET Concurrency Setting

By default, ASP.NET limits request concurrency to reduce steady-state memory consumption on the server. High concurrency applications may need to adjust some settings to improve overall performance. These settings are stored under the following registry entry:

HKEY\_LOCAL\_MACHINE\Software\Microsoft\ASP.NET\2.0.50727.0\Parameters

The following setting is useful for fully using resources on a system:

* **MaxConcurrentRequestPerCpu.**Default value 12.

This setting limits the maximum number of concurrently executing ASP.Net requests on a system. The default value is conservative to reduce memory consumption of ASP.NET applications. Applications that perform long, synchronous I/O operations can experience high user-perceived latency because of queuing or request failures from exceeding queue limits under high load with the default setting.

### Worker Process and Recycling Options

The options for recycling IIS worker processes under the IIS Admin user interface provide practical solutions to acute situations or events without requiring intervention, a service reset, or even a computer reset. Such situations and events include memory leaks, increasing memory load, or unresponsive or idle worker processes. Under ordinary conditions, recycling options might not be needed and can be turned off or the system can be configured to recycle very infrequently. In the following sections, bold names are per-application-pool variables.

You can enable process recycling for a particular application by adding attributes to the recycling/periodicRestart element. The recycle event can be triggered by several events including memory usage, a fixed number of requests, and a fixed time period. When a worker process is recycled, the queued and executing requests are drained and a new process is simultaneously started to service new requests.

system.applicationHost/applicationPools/ApplicationPoolDefaults/recycling/periodicRestart

| **Attribute** | **Description** | **Default** |
| --- | --- | --- |
| *memory* | Enable process recycling if virtual memory consumption exceeds the specified limit in megabytes. This is a useful setting for 32-bit machines that have a small, 2‑GB address space to avoid failed requests because of out-of-memory errors. | 0 |
| *privateMemory* | Enable process recycling if private memory allocations exceed a specified limit in megabytes. | 0 |
| *requests* | Enable process recycling after a certain number of requests. | 0 |
| *time* | Enable process recycling after a specified time period. (The default is 29 hours.) | 29:00:00 |

### Secure Sockets Layer Tuning Parameters

The use of secure sockets layer (SSL) imposes additional CPU cost. The most expensive component of SSL is the session establishment cost (involving a full handshake), and then reconnection cost and encryption/decryption cost. For better SSL performance, do the following:

* Enable keep-alives for SSL sessions. This eliminates the session establishment costs.
* Reuse sessions when appropriate, especially with non-keep-alive traffic.
* Note that larger keys provide more security but also use more CPU time.
* Note that not all components of your page might need to be encrypted. However, mixing plain HTTP and HTTPS might result in a pop-up warning on the client browser that not all content on the page is secure.

### ISAPI

No special tuning parameters are needed for the Internet Server API (ISAPI) applications. If writing a private ISAPI extension, make sure that you code it efficiently for performance and resource use. See also “[Other Issues that Affect IIS Performance](#_Other_Issues_Affecting)” later in this guide.

### Managed Code Tuning Guidelines

The new integrated pipeline model in IIS 7.0 enables a high degree of flexibility and extensibility. Custom modules that are implemented in native or managed code can be inserted into the pipeline or can replace existing modules. Although this extensibility model offers convenience and simplicity, you should be careful before you insert new managed modules that hook into global events. Adding a global managed module means that all requests, including static file requests, must touch managed code. Custom modules are susceptible to events such as garbage collection in addition to adding significant CPU cost because of marshaling data between native and managed code. If possible, you should implement global modules in native (C/C++) code.

When you first deploying an ASP.NET Web site, make sure that you precompile all scripts. You can do this by calling one .NET script in each directory. Reset IIS after compilation is complete. Recompile after changes to Machine.config, Web.config, or any .aspx script.

If session state is not needed, make sure that you turn it off for each page.

When you run multiple hosts that contain ASP.NET scripts in isolated mode (one application pool per site), monitor the memory usage. Make sure that the server that runs has enough RAM for the expected number of concurrently running application pools. Consider using multiple application-domains instead of multiple isolated processes.

For performance recommendations on ASP.NET, see the MSDN article “[10 Tips for Writing High-Performance Web Applications](http://go.microsoft.com/fwlink/?LinkId=98290).”

### Other Issues that Affect IIS Performance

The following issues affect IIS performance:

* Installation of filters that are not cache-aware.

The installation of a filter that is not HTTP-cache-aware causes IIS to completely disable caching, which results in poor performance. Old ISAPI filters that were written before IIS 6.0 can cause this behavior.

* Common Gateway Interface (CGI) requests.

For performance reasons, the use of CGI applications for serving requests is not recommended under IIS. The frequent creation and deletion of CGI processes involves significant overhead. Better alternatives include the use of ISAPI application and ASP or ASP.NET scripts. Isolation is available for each of these options.

### NTFS File System Setting

Under HKLM\System\CurrentControlSet\Control\FileSystem\ is NtfsDisableLastAccessUpdate (REG\_DWORD) 1.

This system-global switch reduces disk I/O load and latencies by disabling the updating of the date and time stamp for the last file or directory access. This key is set to 1 by default. You do not need to adjust it on clean installations of Windows Server 2008 by default. Earlier versions of Microsoft operating systems did not have this key set. Disabling the updates is effective when you are using large data sets (or many hosts) that contain thousands of directories. We recommend that you use IIS logging instead if you maintain this information only for Web administration.

**Warning:** Some applications such as incremental backup utilities rely on this update information and no longer function correctly without it.

### Networking Subsystem Performance Settings for IIS

See “[Performance Tuning for Networking Subsystem](#_Performance_Tuning_for)” earlier in this guide.

# Performance Tuning for File Servers

## Selecting the Proper Hardware for Performance

You should select the proper hardware to satisfy the expected file server load, considering average load, peak load, capacity, growth plans, and response times. Hardware bottlenecks limit the effectiveness of software tuning. “[Performance Tuning for Server Hardware](#_Performance_Tuning_for_Networking)” earlier in this guide provides recommendations for hardware. The sections on networking and storage subsystems also apply to file servers.

## Server Message Block Model

The Server Message Block (SMB) model consists of two entities: the client and the server.

On the client, applications perform system calls by requesting operations on remote files. These requests are handled by the redirector subsystem (rdbss.sys) and the SMB mini-redirector (mrxsmb.sys), which translate them into SMB protocol sessions and requests over TCP/IP. Starting with Windows Vista, the SMB 2.0 protocol is supported. The mrxsmb10.sys driver handles legacy SMB traffic, and the mrxsmb20.sys driver handles SMB 2.0 traffic.

On the server, SMB connections are accepted and SMB requests are processed as local file system operations through NTFS and the local storage stack. The srv.sys driver handles legacy SMB traffic, and the srv2.sys driver handles SMB 2.0 traffic. The srvnet.sys component implements the interface between networking and the file server for both SMB protocols. File system metadata and content can be cached in memory through the system cache in the kernel (ntoskrnl.exe).

Figure 6 summarizes the different layers that a user request on a client machine must undergo to perform file operations over the network on a remote SMB file server that uses SMB 2.0.

Figure 6. Windows SMB Components

## Configuration Considerations

Do not enable any services or features that your particular file server and file clients do not require. These might include SMB signing, client-side caching, file system minifilters, search service, scheduled tasks, NTFS encryption, NTFS compression, IPSEC, firewall filters, and antivirus features.

## General Tuning Parameters for Servers

The following registry tuning parameters can affect the performance of file servers:

* **NtfsDisable8dot3NameCreation**

HKLM\System\CurrentControlSet\Control\FileSystem\REG\_DWORD)

The default is 0. This parameter determines whether NTFS generates a short name in the 8.3 (MS‑DOS®) naming convention for long file names and for file names that contain characters from the extended character set. If the value of this entry is 0, files can have two names: the name that the user specifies and the short name that NTFS generates. If the user-specified name follows the 8.3 naming convention, NTFS does not generate a short name.

Changing this value does not change the contents of a file, but it avoids the short-name attribute creation for the file, which also changes how NTFS displays and manages the file. For most file servers, the recommended setting is 1.

* **TreatHostAsStableStorage**

HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters\(REG\_DWORD)

The default is 0. This parameter disables the processing of write flush commands from clients. If the value of this entry is 1, the server performance and client latency for power-protected servers can improve. Workloads that resemble the NetBench file server benchmark benefit from this behavior.

* **AsynchronousCredits**

HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters\(REG\_DWORD)

The default is 512. This parameter limits the number of concurrent “asynchronous” SMB commands that are allowed on a single connection. Some file clients such as IIS servers require a large amount of concurrency, with file change notification requests in particular. The value of this entry can be increased to support these clients.

* **Smb2CreditsMin and Smb2CreditsMax**

HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters\(REG\_DWORD)

The defaults are 64 and 1024, respectively. These parameters allow the server to throttle client operation concurrency dynamically within the specified boundaries. Some clients might achieve increased throughput with higher concurrency limits. One example is file copy over high-bandwidth, high-latency links.

* **PagedPoolSize** (no longer required for Windows Server 2008)

HKLM\System\CurrentControlSet\Control\SessionManager\MemoryManagement  
\(REG\_DWORD)

* **Disablelastaccess** (no longer required for Windows Server 2008)

HKLM\System\CurrentControlSet\Control\FileSystem\(REG\_DWORD)

* **NumTcbTablePartitions** (no longer required for Windows Server 2008)

HKLM\system\CurrentControlSet\Services\Tcpip\Parameters\(REG\_DWORD)

* **TcpAckFrequency** (no longer required for Windows Server 2008)

HKLM\system\CurrentControlSet\Services\Tcpip\Parameters\Interfaces

## General Tuning Parameters for Client Computers

* **DormantFileLimit**

HKLM\system\CurrentControlSet\Services\lanmanworkstation\parameters  
\(REG\_DWORD)

Windows XP client computers only. By default, this registry key is not created.

This parameter specifies the maximum number of files that should be left open on a share after the application has closed the file.

* **ScavengerTimeLimit**

HKLM\system\CurrentControlSet\Services\lanmanworkstation\parameters  
\(REG\_DWORD)

Windows XP client computers only.

This is the number of seconds that the redirector waits before it starts scavenging dormant file handles (cached file handles that are currently not used by any application).

* **DisableByteRangeLockingOnReadOnlyFiles**

HKLM\System\CurrentControlSet\Services\LanmanWorkStation\Parameters

\(REG\_DWORD)

Windows XP client computers only.

Some distributed applications that lock parts of a read-only file as synchronization across clients require that file-handle caching and collapsing behavior be off for all read-only files. This parameter can be set if such applications will not be run on the system and collapsing behavior can be enabled on the client computer.

# Performance Tuning for Active Directory Servers

The performance of Active Directory®, especially in large environments, can be improved by following these tuning steps:

* Increase address space by using 64-bit processors.

For running Active Directory, 64-bit processors are preferred. Their large address space makes it possible to equip the server with enough RAM to cache all or most of the Active Directory database in memory. It also provides room for expansion to add RAM if the database size grows. For more information, see “[Active Directory Performance for 64-bit Versions of Windows Server 2003](http://www.microsoft.com/downloads/details.aspx?FamilyID=52e7c3bd-570a-475c-96e0-316dc821e3e7).”

* Increase user-mode address space on 32-bit x86 servers.

On servers that have 32-bit x86 processors, use the IncreaseUserVA boot option to increase user-mode address space. This increases how much virtual address space is available to Active Directory and lets Active Directory improve its caching. This option can be set by using the bcdedit tool as follows:

bcdedit /set IncreaseUserVA 3072

This option is the equivalent of the /3GB boot.ini option in Windows Server 2003.

* Use an appropriate amount of RAM.

Active Directory uses the server’s RAM to cache as much of the directory database as possible. This reduces disk access and improves performance. Unlike Windows 2000, the Active Directory cache in Windows Server 2003 and Windows Server 2008 is permitted to grow. However, it is still limited by the virtual address space and how much physical RAM is on the server.

To determine whether more RAM is needed for the server, monitor the percentage of Active Directory operations that are being satisfied from the cache by using the Reliability and Performance Monitor. Examine the lsass,exe instance (for Active Directory Domain Services) or Directory instance (for Active Directory Lightweight Directory Services) of the Database\Database Cache % Hit performance counter. A low value indicates that many operations are not being satisfied from the cache. Adding more RAM might improve the cache hit rate and the performance of Active Directory. You should examine the counter after Active Directory has been running for some time under a typical workload. The cache starts out empty when the Active Directory service is restarted or the machine is rebooted, so the initial hit rate is low.

The use of the Database Cache % Hit counter is the preferred way to assess how much RAM a server needs. Or, a guideline is that when the RAM on a server is twice the physical size of the Active Directory database on disk, it likely gives sufficient room for caching the entire database in memory. However, in many scenarios this is an overestimation because the actual part of the database frequently used is only a fraction of the entire database.

* Use a good disk I/O subsystem.

Ideally, the server is equipped with sufficient RAM to be able to cache the “hot” parts of the database entirely in memory. However, the on-disk database must still be accessed to initially populate the memory cache, when it accesses uncached parts of the database and when it writes updates to the directory. Therefore, appropriate selection of storage is also important to Active Directory performance.

We recommend that the Active Directory database folder be located on a physical volume that is separate from the Active Directory log file folder. In the Active Directory Lightweight Directory Services installation wizard, these are known as data files and data recovery files. Both folders should be on a physical volume that is separate from the operating system volume. The use of drives that support command queuing, especially SCSI or Serial Attached SCSI, might also improve performance.

## Considerations for Read-Heavy Scenarios

The typical directory workload consists of more query operations than update operations. Active Directory is optimized for such a workload. To obtain the maximum benefit, the most important performance tuning step is to make sure that the server has sufficient RAM to be able to cache the most frequently used part of the database in memory. Query performance on a freshly rebooted server, or after the Active Directory service has been restarted, might initially be low until the cache is populated. Active Directory automatically populates the cache as queries visit parts of the directory.

## Considerations for Write-Heavy Scenarios

Write-heavy scenarios do not benefit as much from the Active Directory cache. To guarantee the transactional durability of data that is written to the directory, Active Directory does not cache disk writes. It commits all writes to the disk before it returns a successful completion status for an operation, unless explicitly requested not to do this. Therefore, fast disk I/O is important to the performance of writes to Active Directory. The following are hardware recommendations that might improve performance for these scenarios:

* Hardware RAID controllers.
* Low-latency/high-RPM disks.
* Battery-backed write caches on the controller.

To determine whether disk I/O is a bottleneck, monitor the Physical Disk\Average Disk Queue Length counter for the volumes on which the Active Directory database and logs are located. A high queue length indicates a large amount of disk I/O that is being serialized. Choosing a storage system to improve write performance on those volumes might improve Active Directory performance.

## Using Indexing to Increase Query Performance

Indexing of attributes is useful when you search for objects that have the attribute name in the filter. Indexing can reduce the number of objects that must be visited when you evaluate the filter. However, this reduces the performance of write operations because the index must be updated when the corresponding attribute is modified or added. You can use logging (as mentioned in the Knowledge Base article “[How to configure Active Directory diagnostic event logging in Windows Server 2003 and in Windows 2000 Server](http://support.microsoft.com/kb/314980)”) to find the expensive and inefficient queries and consider indexing some attributes that are used in the corresponding queries to improve the search performance.

## Optimizing Trust Paths

Trusts are a way to enable users to authenticate across different forests or domains. If the trust path between the resource and the user is long, then the user might experience high latency because the authentication request must travel through the trust path and return. For example, if a user from the grandchild of a domain tries to log on from a different grandchild in the same forest, the authentication request must travel up the chain from the grandchild to the root and then take the path to the other grandchild. To avoid this, you can create a shortcut trust directly between the two grandchild domains that avoids the long path. However, the administrator must manage trusts. Therefore you must consider how frequently a given trust will be used before you create it. You can create “external trusts” to reduce the trust path when authenticating between inter-forest domains.

## Active Directory Performance Counters

You can use several resources to conduct performance diagnosis of a domain controller that is not performing as expected.

You can use the followingReliability and Performance Monitor (Perfmon) counters to track and analyze a domain controller’s performance:

* If slow write operations or read operations are noticed, check the following disk I/O counters under the Physical Disk category to see whether many queued disk operations exist:

Avg. Disk Queue Length

Avg. Disk Read Queue Length

Avg. Disk Write Queue Length

* If lsass.exe uses lots of physical memory, check the following Database counters under the Database category to see how much memory is used to cache the database For Active Directory Domain Services. These counters are located under the lsass.exe instance, whereas for Active Directory Lightweight Directory Services they are located under the Directory instance:

Database Cache % Hit

Database Cache Size (MB)

* If Isass.exe uses lots of CPU, check Directory Services\ATQ Outstanding Queued Requests to see how many requests are queued at the domain controller. A high level of queuing indicates that requests are arriving at the domain controller faster than they can be processed. This can also lead to a high latency in responding to requests.

Data Collector Sets is another tool that is included with Windows Server 2008 that you can use to see the activity inside the domain controller. On a server on which the Active Directory Domain Services or Active Directory Lightweight Directory Services role has been installed, the collector template can be found in **Reliability and Performance Monitor** under **Reliability and Performance > Data Collector Sets > System > Active Directory Diagnostics**. To start it, click the **Play** icon.

The data is collected for 5 minutes and a report is generated under **Reliability and Performance > Reports > System > Active Directory Diagnostics**. This report contains information about CPU usage by different processes, Lightweight Directory Access Protocol (LDAP) operations, Directory Services operations, Kerberos Key Distribution Center operations, NT LAN Manager (NTLM) authentications, Local Security Authority/Security Account Manager (LSA/SAM) operations, and averages of all the important performance counters. This report identifies the workload that is being placed on the domain controller, identifies the contribution of different aspects of that workload to the overall CPU usage, and locates the source of that workload such as an application sending a high rate of requests to the domain controller. The CPU section of the report indicates whether lsass.exe is the process that is taking highest CPU percentage. If any other process is taking more CPU on a domain controller, you should investigate it.

# Performance Tuning for Terminal Server

## Selecting the Proper Hardware for Performance

In a Terminal Server deployment scenario, the choice of hardware is governed by the application set and how the users exercise it. The key factors that affect the number of users and their experience are CPU, memory, disk, and graphics. Earlier in this guide was a discussion on server hardware guidelines. Although these guidelines still apply in this role, this section contains additional guidelines that are specific to Terminal Server, mostly related to the multiuser environment of Terminal Server.

### CPU Configuration

CPU configuration is conceptually determined by multiplying the required CPU to support a session by the number of sessions that the system is expected to support, while maintaining a buffer zone to handle temporary spikes. Multiple processors and cores can help reduce abnormal CPU congestion situations, which are usually caused by a few overactive threads that are contained by a similar number of cores. Therefore, the more cores on a system, the lower the cushion margin that must be built into the CPU usage estimate, which results in a larger percentage of active load per CPU. One important factor to remember is that doubling the number of CPUs does not double CPU capacity. For more considerations, see “[Performance Tuning for](#_Performance_Tuning_for_Networking) [Server Hardware](#_Performance_Tuning_for_Networking)” earlier in this guide.

### Processor Architecture

In a 32-bit architecture, all system processes share a 2‑GB kernel virtual address space, which limits the maximum number of attainable Terminal Server sessions. Because memory that the operating system allocates across all processes shares the same 2‑GB space, increasing the number of sessions and processes eventually exhausts this resource. Significant improvements have been made in Windows Server 2008 to better manage the 2‑GB address space. Some of these improvements include dynamic reallocation across different internal memory subareas. This reallocation is based on consumption compared to Windows Server 2003, which had static allocation that left some fraction of the 2 GB unused depending on the specifics of the usage scenario. The most important kernel memory areas that affect Terminal Server capacity are system page table entries (PTEs), system cache, and paged pool. Improvements also include reducing consumption in some critical areas such as kernel stacks for threads. Nevertheless, either significant performance degradation or failures can occur if the number of sessions or processes is high. Actual values vary significantly with the usage scenario, but a good watermark is approximately 250 sessions. Using large amounts of memory (greater than 12 GB) also consumes substantial amounts from the 2‑GB space for memory management data structures, which further accentuates the issue.

The 64-bit processor architecture provides a significantly higher kernel virtual address space, which makes it much more suitable for systems that need large amounts of memory. Specifically, the x64 version of the 64-bit architecture is the more workable option for Terminal Server deployments because it provides very small overhead when it runs 32-bit processes. The most significant performance drawback when you migrate to 64-bit architecture is significantly greater memory usage.

### Memory Configuration

It is difficult to predict the memory configuration without knowing the applications that users employ. However, the required amount of memory can be estimated by using the following formula:

TotalMem = OSMem + SessionMem \* NS

OSMem is how much memory the operating system requires to run (such as system binary images, data structures, and so on), SessionMem is how much memory processes running in one session require, and NS is the target number of active sessions. The amount of required memory for a session is mostly determined by the private memory reference set for applications and system processes that are running inside the session. Shared pages (code or data) have little effect because only one copy is present on the system.

One interesting observation is that, assuming the disk system that is backing the pagefile does not change, the larger the number of concurrent active sessions the system plans to support, the bigger the per-session memory allocation must be. If the amount of memory that is allocated per session is not increased, the number of page faults that active sessions generate increases with the number of sessions and eventually overwhelms the I/O subsystem. By increasing the amount of memory that is allocated per session, the probability of incurring page faults decreases, which helps reduce the overall rate of page faults.

### Disk

Storage is one of the aspects most often overlooked when you configure a Terminal Server system, and it can be the most common limitation on systems that are deployed in the field.

The disk activity that is generated on a typical Terminal Server system affects the following three areas:

* System files and application binaries.
* Pagefiles.
* User profiles and user data.

Ideally, these three areas should be backed by distinct storage devices. Using RAID configurations or other types of high-performance storage further improves performance. We highly recommend that you use storage adapters with battery-backed cache that allows writeback optimizations. Controllers with writeback cache support offer improved support for synchronous disk writes. Because all users have a separate hive, synchronous disk writes are significantly more common on a Terminal Server system. Registry hives are periodically saved to disk by using synchronous write operations. To enable these optimizations, from the Disk Management console, open the **Properties** dialog box for the destination disk and, on the **Policies** tab, select the **Enable write caching on the disk** and **Enable advanced performance** check boxes.

For more specific storage tunings, see the guidelines in “[Performance Tuning for Storage Subsystem](#_Performance_Tuning_for_1)” earlier in this guide.

### Network

Network usage includes two main categories:

* Terminal Server connections traffic in which usage is determined almost exclusively by the drawing patterns exhibited by the applications that are running inside the sessions and the redirected devices I/O traffic.

For example, applications handling text processing and data input consume bandwidth of approximately 10 to 100 Kb per second, whereas rich graphics and video playback cause significant increases in bandwidth usage. We do not recommend video playback over Terminal Server connections because desktop remoting is not optimized to support the high frame rate rendering that is associated with video playback. Frequent use of device redirection features such as file, clipboard, printer, or audio redirection also significantly increases network traffic. Generally, a single 1‑GB adapter is satisfactory for most systems.

* Back-end connections such as roaming profiles, application access to file shares, database servers, e-mail servers, and HTTP servers.

The volume and profile of network traffic is specific to each deployment.

## Tuning Applications for Terminal Server

Most of the CPU usage on a Terminal Server system is driven by applications. Desktop applications are usually optimized toward responsiveness with the goal of minimizing how long it takes an application to respond to a user request. However, in a server environment it is equally important to minimize the total amount of CPU that is used to complete an action to avoid adversely affecting other sessions.

Consider the following suggestions when you configure applications to be used on a Terminal Serve system:

* Minimize background/Idle loop processing.

Typical examples are disabling background grammar/spell checking, data indexing for search, and background saves.

* Minimize how often an application polls to do a state check or update.

Disabling such behaviors or increasing the interval between polling iterations and timer firing significantly benefits CPU usage because the CPU effect of such activities is quickly amplified for many active sessions. Typical examples are connection status icons and status bar information updates.

* Minimize resource contention between applications by reducing their synchronization frequency with that resource.

Examples of such resources include registry keys and configuration files. Examples of such application components and features are status indicator (like shell notifications), background indexing or change monitoring, and offline synchronization.

* Disable unnecessary processes that are registered to be started at user logon or session startup.

These processes can significantly contribute to the CPU cost of creating a new session for the user, which generally is a CPU-intensive process and can be very expensive in morning scenarios. Use MsConfig.exe or MsInfo32.exe to obtain a list of processes that are started at user logon.

* When possible, avoid multimedia application components for Terminal Server deployments.

Video playback causes high bandwidth usage for the Terminal Server connection, and audio playback causes high bandwidth usage on the audio redirection channel. Also, multimedia processing (encoding and decoding, mixing, and so on) has a significant CPU usage cost.

For memory consumption, consider the following suggestions:

* Verify that dlls that applications load are not relocated at load.

If dlls are relocated, it is impossible to share their code across sessions, which significantly increases the footprint of a session. This is one of the most common memory-related performance problems in Terminal Server.

* For common language runtime (CLR) applications, use Native Image Generator (Ngen.exe) to increase page sharing and reduce CPU overhead.

When possible, apply similar techniques to other similar execution engines.

## Terminal Server Tuning Parameters

### Pagefile

Insufficient pagefile can cause memory allocation failures either in applications or system components. A general guideline is that the combined size of the pagefiles should be two to three times larger than the physical memory size. You can use the Memory\Committed Bytes performance counter to monitor how much committed virtual memory is on the system. When the value of this counter reaches close to the total combined size of physical memory and pagefiles, memory allocation begins to fail. Because of significant disk I/O activity that pagefile access generates, consider using a dedicated storage device for the pagefile, ideally a high-performance one such as a RAID array.

### Antivirus and Antispyware

Installing antivirus and antispyware software on a Terminal Server greatly affects overall system performance, especially on CPU usage. We highly recommend that you exclude from the active monitoring list all the folders that hold temporary files, especially those that services and other system components generate. Generally, antispyware software has a much more significant performance effect than antivirus software does and should be installed only when it is necessary.

### Task Scheduler

Task Scheduler (which can be accessed under **All Programs** > **Accessories** > **System Tools**) lets you examine the list of tasks that are scheduled for different events. For Terminal Server, it is useful to focus specifically on the tasks that are configured to run on idle, at user logon, or on session connect and disconnect. Because of the specifics assumptions of the deployment, many of these tasks might be unnecessary.

### Desktop Notification Icons

Notification icons on the Desktop can have fairly expensive refreshing mechanisms. You can use **Customize Notifications Icons** to examine the list of notifications that are available in the system. Generally, it is best to disable unnecessary notifications by either removing the component that registers them from the startup list or by changing the configuration on applications and system components to disable them.

You can implement the following tuning parameters by opening the MMC snap-in for Group Policy (gpedit.smc) and making the respective changes under **Computer Configuration** > **Administrative Templates** > **Windows Components** > **Terminal Services** > **Terminal Server**:

* Color depth.

Color depth can be adjusted under **Remote Session Environment** > **Limit Maximum Color Depth** with possible values of 8, 15, 16, and 32 bit. The default value is 16 bit, and increasing the bit depth increases memory and bandwidth consumption. Or, the color depth can be adjusted from TSConfig.exe by opening the **Properties** dialog box for a specific connection and, on the **Client Setting** tab, changing the selected value in the drop-down box under **Color Depth**. The **Limit Maximum Color Depth** check box must be selected.

* Remote Desktop Protocol compression.

Remote Desktop Protocol (RDP) compression can be configured under **Remote Session Environment** > **Set compression algorithm for RDP data**. Three values are possible:

**Optimized to use less memory** is the configuration that matches the default Windows Server 2003 configuration. This uses the least amount of memory per session but has the lowest compression ratio and therefore the highest bandwidth consumption.

**Balances memory and network bandwidth** is the default setting for Windows Server 2008. This has reduced bandwidth consumption while marginally increasing memory consumption (approximately 200 KB per session).

**Optimized to use less network bandwidth** further reduces network bandwidth usage at a cost of approximately 2 MB per session. This memory is allocated in the kernel virtual address space and can have a significant effect on 32-bit processor-based systems that are running a fairly large number of users. Because 64-bit systems do not have these issues, this setting is recommended if the additional memory cost is considered acceptable. If you want to use this setting, you should assess the maximum number of sessions and test to that level with this setting before placing a server in production.

* Device redirection.

Device redirection can be configured under **Device and Resource Redirection**. Or, it can be configured through TSConfig by opening the properties for a specific connection such as **RDP-Tcp** and, on the **Client Settings** tab, changing **Redirection** settings.

Generally, device redirection increases how much network bandwidth Terminal Server connections use because data is exchanged between devices on the client machines and processes that are running in the server session. The extent of the increase is a function of the nature of frequency of operations that are performed by the applications that are running on the server against the redirected devices.

Printer redirection and Plug and Play device redirection also increase logon CPU usage. You can redirect printers in two ways:

Matching printer driver-based redirection when a driver for the printer must be installed on the server. Earlier releases of Windows Server used this method.

Easy Print printer driver redirection, which is a new method in Windows Server 2008 that uses a common printer driver for all printers.

We recommend the Easy Print method because it causes less CPU usage for printer installation at connection time. The matching driver method causes increased CPU usage because it requires the spooler service to load different drivers. For bandwidth usage, the Easy Print method causes slightly increased network bandwidth usage, but not significant enough to offset the other performance, manageability, and reliability benefits.

Audio redirection is disabled by default because using it causes a steady stream of network traffic. Audio redirection also enables users to run multimedia applications that typically have high CPU consumption.

### Client Experience Settings

The Terminal Server Client provides control over a range of settings that influence network bandwidth performance for the Terminal Server connection. You can access them either through the Terminal Server Client user interface on the **Experience** tab or as settings in the RDP file:

* **Disable wallpaper** (RDP file setting: disable wallpaper:i:0) suppresses the display of desktop wallpaper on redirected connections. It can significantly reduce bandwidth usage if desktop wallpaper consists of an image or other content with significant drawing cost.
* **Font smoothing** (RDP file setting: allow font smoothing:i:0) controls ClearType font rendering support. Although this improves the rendering quality for fonts when it is enabled, it does affect network bandwidth consumption significantly (generally more than 400 percent).
* **Desktop composition** is supported only for a remote session to Windows Vista and has no relevance for server systems.
* **Show contents of windows while dragging** (RDP file setting: disable full window drag:i:1), when it is disabled, reduces bandwidth by displaying only the window frame instead of all the contents when dragged.
* **Menu and window animation** (represented by two distinct RDP file settings: disable menu anims:i:1 and disable cursor setting:i:1), when it is disabled, reduces bandwidth by disabling animation on menus (such as fading) and cursors.
* **Themes** (RDP file setting: disable themes:i:1), when it is disabled, reduces bandwidth by simplifying theme drawings that use the classic theme.
* **Bitmap cache** (RDP file setting: bitmapcachepersistenable:i:1), when it is enabled, creates a client-side cache of bitmaps that are rendered in the session. It is a significant improvement on bandwidth usage and should always be enabled (except for security considerations).

## Desktop Size

Desktop size for remote sessions can be controlled either through the TS Client user interface (on the **Display** tab under **Remote desktop size** settings) or the RDP file (desktopwidth:i:1152 and desktopheight:i:864). The larger the desktop size, the greater the memory and bandwidth consumption that is associated with that session. The current maximum desktop size that a server accepts is 4096 x 2048.

## Windows System Resource Manager

Windows System Resource Manager (WSRM) is an optional component that is available in Windows Server 2008 that now supports an “equal per session” built-in policy that keeps CPU usage equally distributed among all active sessions on the system. Although enabling WSRM adds some CPU usage overhead to the system, the advantage is that it helps limit the effect that high CPU usage in one session has on the other sessions on the system. This helps improve user experience and also lets you run more users on the system because of a reduced need for a large cushion in CPU capacity to accommodate random CPU usage spikes.

# Performance Tuning for Terminal Server Gateway

This section describes the performance-related parameters that help improve the performance of a customer deployment and the tunings that rely on their network usage patterns. At its core, the TS Gateway performs many packet forwarding operations between the TS Client instances and the TS Server instances within the customer’s network. The IIS and TS Gateway export the following registry parameters to help improve system performance in the TS Gateway role:

* Thread tunings.

**MaxIoThreads**

HKLM\Software\Microsoft\Terminal Server Gateway\ (REG\_DWORD)

The default value is 5. It specifies the number of threads that the TS Gateway service creates to handle incoming requests.

**MaxPoolThreads**

HKLM\System\CurrentControlSet\Services\InetInfo\Parameters\(REG\_DWORD)

The default value is 4. It specifies the number of Internet Information Services (IIS) pool threads to create per processor. The IIS pool threads watch the network for requests and process all incoming requests. The **MaxPoolThreads** count does not include threads that the TS Gateway service consumes.

* Remote procedure call tunings for TS Gateways.

The following parameters can help tune the remote procedure call (RPC) receive windows on the TS Client and TS Gateway machines. Changing the windows helps throttle how much data is flowing through each connection and can improve performance for RPC over HTTP v2 scenarios.

**ServerReceiveWindow**

HKLM\Software\Microsoft\Rpc\ (REG\_DWORD)

The default value is 64 KB. This value specifies the receive window that the server uses for data that is received from the RPC proxy. The minimum value is set to 8 KB, and the maximum value is set at 1 GB. If the value is not present, then the default value is used. When changes are made to this value, IIS must be restarted for the change to take effect.

**ClientReceiveWindow**

HKLM\Software\Microsoft\Rpc\ (REG\_DWORD)

The default value is 64 KB. This value specifies the receive window that the client uses for data that is received from the RPC proxy. The minimum valid value is 8 KB, and the maximum value is 1 GB. If the value is not present, then the default value is used.

## Monitoring and Data Collection

The following list of performance counters is considered a base set of counters when you monitor the resource usage on the Terminal Server Gateway:

\Terminal Service Gateway\\*  
\RPC/HTTP Proxy\\*  
\RPC/HTTP Proxy Per Server\\*  
\Web Service\\*  
\W3SVC\_W3WP\\*  
\IPv4\\*  
\Memory\\*  
\Network Interface(\*)\\*  
\Process(\*)\\*  
\Processor(\*)\\*  
\System\\*  
\TCPv4\\*

**Note**: If applicable, add the “\IPv6\\*” and “\TCPv6\\*” objects.

# Performance Tuning for Virtualization Servers

Hyper‑V is the new virtualization server role in Windows Server 2008. Virtualization servers can host multiple virtual machines (VMs), which are isolated from each other but share the underlying hardware resources by virtualizing the processors, memory, and I/O devices. By consolidating servers onto a single machine, virtualization can improve resource usage and power efficiency and reduce the operational and maintenance costs of servers. In addition, VMs and the management APIs offer more flexibility for managing resources, balancing load, and provisioning systems.

This section provides terminology that is used throughout the text and suggests best practices that yield increased performance on Hyper‑V servers.

## Terminology

This section summarizes key terminology specific to VM technology that will be used throughout this performance tuning guide:

child partition

Any partition (VM) that is created by the root partition.

device virtualization

A mechanism that lets a hardware resource be abstracted and shared among multiple consumers.

emulated device

A virtualized device that mimics an actual physical hardware device so that guests can use the typical drivers for that hardware device.

enlightenment

An optimization to a guest operating system to make it aware of VM environments and tune its behavior for VMs.

guest

Software that is running in a partition. It can be a full-featured operating system or a small, special-purpose kernel. The hypervisor is “guest-agnostic.”

hypervisor

A layer of software that sits just above the hardware and below one or more operating systems. Its primary job is to provide isolated execution environments called partitions. Each partition has its own set of hardware resources (CPU, memory, and devices). The hypervisor is responsible for controls and arbitrates access to the underlying hardware.

logical processor

A CPU that handles one thread of execution (instruction stream). There can be one or more logical processors per core and one or more cores per processor socket. In effect, it is a “physical processor.”

passthrough disk access

A representation of an entire physical disk as a virtual disk within the guest. The data and commands are “passed through” to the physical disk (through the root partition’s native storage stack) with no intervening processing by the virtual stack.

root partition

A partition that is created first and owns all the resources that the hypervisor does not own including most devices and system memory. It hosts the virtualization stack and creates and manages the child partitions.

synthetic device

A virtualized device with no physical hardware analog so that guests might need a driver (virtualization service client) to that synthetic device. The driver can use VMBus to communicate with the virtualized device software in the root partition.

virtual machine (VM)

A virtual computer that was created by software emulation and has the same characteristics as a real computer.

virtual processor

A virtual abstraction of a processor that is scheduled to run on a logical processor. A VM can have one or more virtual processors.

virtualization service client (VSC)

A software module that a guest loads to consume a resource or service. For I/O devices, the virtualization service client can be a device driver that the operating system kernel loads.

virtualization service provider (VSP)

A provider, exposed bythe virtualization stack, that provides resources or services such as I/O to a child partition.

virtualization stack

A collection of software components in the root partition that work together to support VMs. The virtualization stack works with and sits above the hypervisor. It also provides management capabilities.

## Hyper‑V Architecture

Hyper‑V features a hypervisor-based architecture that is shown in Figure 7. The hypervisor virtualizes processors and memory and provides mechanisms for the virtualization stack in the root partition to manage child partitions (VMs) and expose services such as I/O devices to the VMs. The root partition owns and has direct access to the physical I/O devices. The virtualization stack in the root partition provides a memory manager for VMs, management APIs, and virtualized I/O devices. It also implements emulated devices such as Integrated Device Electronics (IDE) and PS/2 but supports synthetic devices for increased performance and reduced overhead.

**Root Partition**

I/O  
Stack

Drivers

**Child Partition**

I/O  
Stack

VSCs

Server

**Child Partition**

I/O  
Stack

VSCs

Server

**Hypervisor**

**Devices**

**Processors**

**Memory**

VMBus

VMBus

VMBus

Shared Memory

VSPs

VSPs

OS Kernel Enlightenments (WS08+)

Figure 7. Hyper‑V Hypervisor-Based Architecture Diagram

The synthetic I/O architecture consists of VSPs in the root partition and VSCs in the child partition. Each service is exposed as a device over VMBus, which acts as an I/O bus and enables high-performance communication between VMs that use mechanisms such as shared memory. Plug and Play enumerates these devices, including VMBus, and loads the appropriate device drivers (VSCs). Services other than I/O are also exposed through this architecture.

Windows Server 2008 features enlightenments to the operating system to optimize its behavior when it is running in VMs. The benefits include reducing the cost of memory virtualization, improving multiprocessor scalability, and decreasing the background CPU usage of the guest operating system.

## Server Configuration

This section describes best practices for selecting hardware for virtualization servers and installing and setting up Windows Server 2008 for the Hyper‑V server role.

### Hardware Selection

The hardware considerations for Hyper‑V servers generally resemble that of servers , but Hyper‑V servers can exhibit increased CPU usage, consume more memory, and need larger I/O bandwidth because of server consolidation. For more information, refer to “[Performance Tuning for Server Hardware](#_Performance_Tuning_for_Networking)” earlier in this guide.

* Processors.

Hyper‑V in Windows Server 2008 supports up to 16 logical processors and can use all logical processors if the number of active virtual processors matches that of logical processors. This can reduce the rate of context switching between virtual processors and can yield better performance overall.

* Cache.

Hyper‑V can benefit from larger processor caches, especially for loads that have a large working set in memory and in VM configurations in which the ratio of virtual processors to logical processors is high.

* Memory.

The physical server requires sufficient memory for the root and child partitions. Hyper‑V first allocates the memory for child partitions, which should be sized based on the needs of the expected server load for each VM. The root partition should have sufficient available memory to efficiently perform I/Os on behalf of the VMs and operations such as a VM snapshot.

* Networking.

If the expected loads are network intensive, the virtualization server can benefit from having multiple network adapters or multiport network adapters. VMs can be distributed among the adapters for better overall performance. To reduce the CPU usage of network I/Os from VMs, Hyper‑V can use hardware offloads such as Large Send Offload (LSOv1) and TCPv4 checksum offload. For details on network hardware considerations, see “[Performance Tuning for Networking Subsystem](#_Performance_Tuning_for)” earlier in this guide.

* Storage.

The storage hardware should have sufficient I/O bandwidth and capacity to meet current and future needs of the VMs that the physical server hosts. Consider these requirements when you select storage controllers and disks and choose the RAID configuration. Placing VMs with highly disk-intensive workloads on different physical disks will likely improve overall performance. For example, if four VMs share a single disk and actively use it, each VM can yield only 25 percent of the bandwidth of that disk. For details on storage hardware considerations and discussion on sizing and RAID selection, see “[Performance Tuning for Storage Subsystem](#_Performance_Tuning_for_1)” earlier in this guide.

### Server Core Installation Option

Windows Server 2008 features the Server Core installation option. Server Core offers a minimal environment for hosting a select set of server roles including Hyper‑V. It features a smaller disk, memory profile, and attack surface. Therefore, we highly recommend that Hyper‑V virtualization servers use the Server Core installation option. Using Server Core in the root partition leaves additional memory for the VMs to use (approximately 80 MB for commit charge on 64-bit Windows).

Server Core offers a console window only when the user is logged on, but Hyper‑V exposes management features through WMI so administrators can manage it remotely (see [MSDN](http://msdn2.microsoft.com/en-us/library/cc136992(VS.85).aspx)).

### Dedicated Server Role

The root partition should be dedicated to the virtualization server role. Additional server roles can adversely affect the performance of the virtualization server, especially if they consume significant CPU, memory, or I/O bandwidth. Minimizing the server roles in the root partition has additional benefits such as reducing the attack surface and the frequency of updates.

System administrators should consider carefully what software is installed in the root partition because some software can adversely affect the overall performance of the virtualization server.

### Guest Operating Systems

Hyper‑V supports and has been tuned for both 32-bit and 64-bit versions of Windows Server 2008 and Windows Server 2003 (SP2 or later versions required) as guest operating systems. The number of virtual processors that are supported per guest depends on the guest operating system. Windows Server 2008 is supported with 1P, 2P, and 4P VMs, and Windows Server 2003 SP2 is supported with 1P and 2P VMs. For the list of other supported guest operating systems, see the documentation that is provided with the Hyper‑V installation.

The VM integration services, which significantly improve performance, might not work on unsupported guest operating systems.

### CPU Statistics

Hyper‑V publishes performance counters to help characterize the behavior of the virtualization server and break out the resource usage. The standard set of tools for viewing performance counters in Windows include Performance Monitor (perfmon.exe) and logman.exe, which can display and log the Hyper‑V performance counters. The names of the relevant counter objects are prefixed with “Hyper‑V.”

You should always measure the CPU usage of the physical system through the Hyper‑V Hypervisor Logical Processor performance counters. The statistics that Task Manager and Performance Monitor report in the root and child partitions do not fully capture the CPU usage.

## Processor Performance

The hypervisor virtualizes the physical processors by time-slicing between the virtual processors. To perform the required emulation, certain instructions and operations require the hypervisor and virtualization stack to run. Migrating a workload into a VM increases the CPU usage, but this guide describes best practices for minimizing that overhead.

### Integration Services

The VM integration services include enlightened drivers for the synthetic I/O devices, which significantly reduces CPU overhead for I/O than for emulated devices. The latest version should be installed in every supported guest. The services decrease the CPU usage of the guests, from idle guests to heavily used guests, and improve the I/O throughput. This is the first step in tuning a Hyper‑V server for performance.

### Enlightened Guests

The operating system kernel in Windows Vista SP1, Windows Server 2008, and later releases features enlightenments that optimize its operation for VMs. For best performance, we recommend that you use Windows Server 2008 as a guest operating system. The enlightenments decrease the CPU overhead of Windows that runs in a VM. The integration services provide additional enlightenments for I/O. Depending on the server load, it can be appropriate to host a server application in a Windows Server 2008 guest for better performance.

### Virtual Processors

Hyper‑V in Windows Server 2008 supports a maximum of four virtual processors per VM. VMs that have loads that are not CPU intensive should be configured by using one virtual processor. This is because of the additional overhead that is associated with multiple virtual processors, such as additional synchronization costs in the guest operating system. More CPU-intensive loads should be placed in 2P or 4P VMs if the VM requires more than one CPU of processing under peak load.

Hyper‑V supports Windows Server 2008 guests in 1P, 2P, or 4P VMs, and Windows Server 2003 supports SP2 guests in 1P and 2P VMs. Windows Server 2008 features enlightenments to the core operating system that improves scalability in multiprocessor VMs. Your workloads can benefit from the scalability improvements in Windows Server 2008 if they must run 2P and 4P VMs.

### Background Activity

Minimizing the background activity in idle VMs releases CPU cycles that can be used elsewhere by other VMs or saved to reduce power consumption. Windows guests typically use less than 1 percent of one CPU when they are idle. The following are several best practices for minimizing the background CPU usage of a VM:

* Install the latest version of VM integration services.
* Remove the emulated network adapter through the VM settings dialog box (use a synthetic adapter).
* Disable the screen saver or select a blank screen saver.
* Remove unused devices such as the CD-ROM and COM port, or disconnect their media.
* Keep the Windows guest at the logon screen when it is not being used (and disable its screen saver).
* Use Windows Server 2008 for the guest operating system.
* Disable, throttle, or stagger periodic activity such as backup and defragmentation if appropriate.
* Review scheduled tasks and services enabled by default.
* Improve server applications to reduce periodic activity (such as timers).

The following are additional best practices for configuring a *client version* of Windows in a VM to reduce the overall CPU usage:

* Disable background services such as SuperFetch and Windows Search.
* Disable scheduled tasks such as Scheduled Defrag.
* Disable AeroGlass and other user interface effects (through the System application in Control Panel).

### Weights and Reserves

Hyper‑V supports setting the weight of a virtual processor to grant it a larger or smaller share of CPU cycles than average and specifying the reserve of a virtual processor to make sure that it gets a minimal percentage of CPU cycles. The CPU that a virtual processor consumes can also be limited by specifying usage limits. System administrators can use these features to prioritize specific VMs, but we recommend the default values unless you have a compelling reason to alter them.

Weights and reserves prioritize or de-prioritize specific VMs if CPU resources are overcommitted. This makes sure that those VMs receive a larger or smaller share of the CPU. Highly intensive loads can benefit from adding more virtual processors instead, especially when they are close to saturating an entire physical CPU.

## Memory Performance

The hypervisor virtualizes the guest physical memory to isolate VMs from each other and provide a contiguous, zero-based memory space for each guest operating system. Memory virtualization can increase the CPU cost of accessing memory, especially when applications frequently modify the virtual address space in the guest operating system because of frequent allocations and deallocations.

### Enlightened Guests

Windows Server 2008 includes kernel enlightenments and optimizations to the memory manager to reduce the CPU overhead from Hyper‑V memory virtualization. Workloads that have a large working set in memory can benefit from using Windows Server 2008 as a guest. These enlightenments reduce the CPU cost of context switching between processes and accessing memory. Additionally, they improve the multiprocessor (MP) scalability of Windows Server 2008 guests.

### Correct Memory Sizing

You should size VM memory as you typically do for server applications on a physical machine. You must size it to reasonably handle the expected load at ordinary and peak times because insufficient memory can significantly increase response times and CPU or I/O usage. In addition, the root partition must have sufficient memory (leave at least 512 MB available) to provide services such as I/O virtualization, snapshot, and management to support the child partitions.

A good standard for the memory overhead of each VM is 32 MB for the first 1 GB of virtual RAM plus another 8 MB for each additional GB of virtual RAM. This should be factored in the calculations of how many VMs to host on a physical server. The memory overhead varies depending on the actual load and amount of memory that is assigned to each VM.

## Storage I/O Performance

Hyper‑V supports synthetic and emulated storage devices in VMs, but the synthetic devices generally can offer significantly better throughput and response times and reduced CPU overhead. The exception is if a filter driver can be loaded and reroutes I/Os to the synthetic storage device. Virtual hard disks (VHDs) can be backed by three types of VHD files or raw disks. This section describes the different options and considerations for tuning storage I/O performance.

For more information, refer to “[Performance Tuning for Storage Subsystem](#_Performance_Tuning_for_1)” earlier in this guide, which discusses considerations for selecting and configuring storage hardware.

### Synthetic SCSI Controller

The synthetic storage controller provides significantly better performance on storage I/Os with reduced CPU overhead than the emulated IDE device. The VM integration services include the enlightened driver for this storage device and are required for the guest operating system to detect it. The operating system disk must be mounted on the IDE device for the operating system to boot correctly, but the VM integration services load a filter driver that reroutes IDE device I/Os to the synthetic storage device.

We strongly recommend that you mount the data drives directly to the synthetic SCSI controller because that configuration has reduced CPU overhead. You should also mount log files and the operating system paging file directly to the synthetic SCSI controller if their expected I/O rate is high.

For highly intensive storage I/O workloads that span multiple data drives, each VHD should be attached to a separate synthetic SCSI controller for better overall performance. In addition, each VHD should be stored on separate physical disks.

### Virtual Hard Disk Types

There are three types of VHD files. We recommend that production servers use fixed-sized VHD files for better performance and also to make sure that the virtualization server has sufficient disk space for expanding the VHD file at run time. The following are the performance characteristics and trade-offs between the three VHD types:

* Dynamically expanding VHD.

Space for the VHD is allocated on demand. The blocks in the disk start as zeroed blocks but are not backed by any actual space in the file. Reads from such blocks return a block of zeros. When a block is first written to, the virtualization stack must allocate space within the VHD file for the block and then update the metadata. This increases the number of necessary disk I/Os for the write and causes an increased CPU usage. Reads and writes to existing blocks incur both disk access and CPU overhead when looking up the blocks’ mapping in the metadata.

* Fixed-size VHD.

Space for the VHD is first allocated when the VHD file is created. This type of VHD is less apt to fragment, which reduces the I/O throughput when a single I/O is split into multiple I/Os. It has the lowest CPU overhead of the three VHD types because reads and writes do not need to look up the mapping of the block.

* Differencing VHD.

The VHD points to a parent VHD file. Any writes to blocks never written to before result in space being allocated in the VHD file, as with a dynamically expanding VHD. Reads are serviced from the VHD file if the block has been written to. Otherwise, they are serviced from the parent VHD file. In both cases, the metadata is read to determine the mapping of the block. Reads and writes to this VHD can consume more CPU and result in more I/Os than a fixed-sized VHD.

Snapshots of a VM create a differencing VHD to store the writes to the disks since the snapshot was taken. Having only a few snapshots can elevate the CPU usage of storage I/Os, but might not noticeably affect performance except in highly I/O-intensive server workloads.

However, having a large chain of snapshots can noticeably affect performance because reading from the VHD can require checking for the requested blocks in many differencing VHDs. Keeping snapshot chains short is important for maintaining good disk I/O performance.

### Passthrough Disks

The VHD in a VM can be mapped directly to a physical disk or logical unit number (LUN), instead of a VHD file. The benefit is that this configuration bypasses the file system (NTFS) in the root partition, which reduces the CPU usage of storage I/O. The risk is that physical disk or LUNs can be more difficult to move between machines than VHD files.

Large data drives can be prime candidates for passthrough disks, especially if they are I/O intensive. VMs that can be migrated between virtualization servers (such as quick migration) must also use drives that reside on a LUN of a shared storage device.

### Disabling File Last Access Time Check

Windows Server 2003 and earlier Windows operating systems update the last-accessed time of a file when applications open, read, or write to the file. This increases the number of disk I/Os, which further increases the CPU overhead of virtualization. If applications do not use the last-accessed time on a server, system administrators should consider setting this registry key to disable these updates.

NTFSDisableLastAccessUpdate

HKLM\System\CurrentControlSet\Control\FileSystem\ (REG\_DWORD)

By default, both Windows Vista and Windows Server 2008 disable the last-access time updates.

### Physical Disk Topology

VHDs that I/O-intensive VMs use generally should not be placed on the same physical disks because the disks can otherwise become a bottleneck. If possible, they should also not be placed on the same physical disks that the root partition uses. For a discussion on capacity planning for storage hardware and RAID selection, see “[Performance Tuning for Storage Subsystem](#_Performance_Tuning_for_1)” earlier in this guide.

### I/O Balancer Controls

The virtualization stack balances storage I/O streams from different VMs so that each VM has similar I/O response times when the system’s I/O bandwidth is saturated. The following registry keys can be used to adjust the balancing algorithm, but the virtualization stack tries to fully use the I/O device’s throughput while providing reasonable balance. The first path should be used for storage scenarios, and the second path should be used for networking scenarios:

HKLM\System\CurrentControlSet\Services\StorVsp\<Key> = (REG\_DWORD)

HKLM\System\CurrentControlSet\Services\VmSwitch\<Key> = (REG\_DWORD)

Both storage and networking have three registry keys at the preceding StorVsp and VmSwitch paths, respectively. Each value is a DWORD and operates as follows. We do not recommend this advanced tuning option unless you have a specific reason to use it. Note that *these registry keys might be removed in future releases:*

* **IOBalance\_Enabled**

The balancer is enabled when set to a nonzero value and disabled when set to 0. The default is enabled for storage and disabled for networking. Enabling the balancing for networking can add significant CPU overhead in some scenarios.

* **IOBalance\_KeepHwBusyLatencyTarget\_Microseconds**

This controls how much work, represented by a latency value, the balancer allows to be issued to the hardware before throttling to provide better balance. The default is 83 ms for storage and 2 ms for networking. Lowering this value can improve balance but will reduce some throughput. Lowering it too much significantly affects overall throughput. Storage systems with high throughput and high latencies can show added overall throughput with a higher value for this parameter.

* **IOBalance\_AllowedPercentOverheadDueToFlowSwitching**

This controls how much work the balancer issues from a VM before switching to another VM. This setting is primarily for storage where finely interleaving I/Os from different VMs can increase the number of disk seeks. The default is 8 percent for both storage and networking.

## Network I/O Performance

Hyper‑V supports synthetic and emulated network adapters in the VMs, but the synthetic devices offer significantly better performance and reduced CPU overhead. Each of these adapters is connected to a virtual network switch, which can be connected to a physical network adapter if external network connectivity is needed.

For how to tune the network adapter in the root partition, including interrupt moderation, refer to “[Performance Tuning for Networking Subsystem](#_Performance_Tuning_for)” earlier in this guide. The TCP tunings in that section should be applied, if required, to the child partitions.

### Synthetic Network Adapter

Hyper‑V features a synthetic network adapter that is designed specifically for VMs to achieve significantly reduced CPU overhead on network I/O when it is compared to the emulated network adapter that mimics existing hardware. The synthetic network adapter communicates between the child and root partitions over VMBus by using shared memory for more efficient data transfer.

The emulated network adapter should be removed through the VM settings dialog box and replaced with a synthetic network adapter. The guest requires that the VM integration services be installed.

### Offload Hardware

As with the native scenario, offload capabilities in the physical network adapter reduce the CPU usage of network I/Os in VM scenarios. Hyper‑V currently uses LSOv1 and TCPv4 checksum offload. The offload capabilities must be enabled in the driver for the physical network adapter in the root partition. For details on offload capabilities in network adapters, refer to “[Choosing a Network Adapter](#_Choosing_a_Network)” earlier in this guide.

Drivers for certain network adapters disable LSOv1 but enable LSOv2 by default. System administrators must explicitly enable LSOv1 by using the driver **Properties** dialog box in Device Manager.

### Network Switch Topology

Hyper‑V supports creating multiple virtual network switches, each of which can be attached to a physical network adapter if needed. Each network adapter in a VM can be connected to a virtual network switch. If the physical server has multiple network adapters, VMs with network-intensive loads can benefit from being connected to different virtual switches to better use the physical network adapters.

### Interrupt Affinity

Under certain workloads, binding the device interrupts for a single network adapter to a single logical processor can improve performance for Hyper‑V. We recommend this advanced tuning only to address specific problems in fully using network bandwidth. System administrators can use the [IntPolicy](http://www.microsoft.com/whdc/system/sysperf/intpolicy.mspx) tool to bind device interrupts to specific processors.

### VLAN Performance

The Hyper‑V synthetic network adapter supports VLAN tagging. It provides significantly better network performance if the physical network adapter supports NDIS\_ENCAPSULATION\_IEEE\_802\_3\_P\_AND\_Q\_IN\_OOB encapsulation for both large send and checksum offload. Without this support, Hyper‑V cannot use hardware offload for packets that require VLAN tagging and network performance can be decreased.

# Performance Tuning for File Server Workload (NetBench)

NetBench 7.02 is an eTesting Labs workload that measures the performance of file servers as they handle network file requests from clients. NetBench gives you an overall I/O throughput score and average response time for your server and with individual scores for the client computers. You can use these scores to measure, analyze, and predict how well your server can handle file requests from clients.

To make sure of a fresh start, the data volumes should always be formatted between tests to flush and clean up the working set. For improved performance and scalability, we recommend that client data be partitioned over multiple data volumes. The networking, storage, and interrupt affinity sections contain additional tuning information that might apply to specific hardware.

## Registry Tuning Parameters for Servers

The following registry tuning parameters can affect the performance of file servers:

* **NtfsDisable8dot3NameCreation**

HKLM\System\CurrentControlSet\Control\FileSystem\ (REG\_DWORD)

The default is 0. This parameter determines whether NTFS generates a short name in the 8.3 (MS‑DOS) naming convention for long file names and for file names that contain characters from the extended character set. If the value of this entry is 0, files can have two names: the name that the user specifies and the short name that NTFS generates. If the name that the user specifies follows the 8.3 naming convention, NTFS does not generate a short name.

Changing this value does not change the contents of a file, but it avoids the short-name attribute creation for the file and also changes how NTFS displays and manages the file. For most file servers, the recommended setting is 1.

* **TreatHostAsStableStorage**

HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters\(REG\_DWORD)

The default is 0. This parameter disables the processing of write flush commands from clients. If the value of this entry is 1, the server performance and client latency for power-protected servers can improve.

## Registry Tuning Parameters for Client Computers

* **DormantFileLimit**

HKLM\system\CurrentControlSet\Services\lanmanworkstation\parameters\REG\_DWORD)

Windows XP client computers only.

This parameter specifies the maximum number of files that should be left open on a share after the application has closed the file.

* **ScavengerTimeLimit**

HKLM\system\CurrentControlSet\Services\lanmanworkstation\parameters\REG\_DWORD)

Windows XP client computers only.

This parameter is the number of seconds that the redirector waits before it starts scavenging dormant file handles (cached file handles that are currently not used by any application).

* **DisableByteRangeLockingOnReadOnlyFiles**

HKLM\System\CurrentControlSet\Services\LanmanWorkStation\Parameters\REG\_DWORD)

Windows XP client computers only.

Some distributed applications that lock parts of a read-only file as synchronization across clients require that file-handle caching and collapsing behavior be off for all read-only files. This parameter can be set if such applications will not be run on the system and collapsing behavior can be enabled on the client computer.

# Performance Tuning for Network Workload ([NTttcp](http://www.microsoft.com/whdc/device/network/TCP_tool.mspx))

## Tuning for NTttcp

NTttcp is a Winsock-based port of [ttcp](http://en.wikipedia.org/wiki/Ttcp) to Windows. It helps measure network driver performance and throughput on different network topologies and hardware setups. It provides the customer a multithreaded, asynchronous performance workload for measuring achievable data transfer rate on an existing network setup.

Options include the following:

* A single thread should be sufficient for optimal throughput.
* Multiple threads are required only for single to many clients.
* Posting enough user receive buffers (by increasing the value passed to the “-a” option) reduces TCP copying.
* You should not excessively post user receive buffers because the first ones that are posted would return before you have the need to use other buffers.
* It is best to bind each set of threads to a processor (the second delimited parameter in the “-m” option).
* Each thread creates a socket that connects (listens) on a different port.

Table 10. Example Syntax for NTttcp Sender and Receiver

|  |  |
| --- | --- |
| Syntax | Details |
| Example Syntax for a Sender  NTttcps –m 1,0,10.1.2.3 –a 2 | Single thread.  Bound to CPU 0.  Connecting to a computer that uses IP 10.1.2.3.  Posting two send overlapped buffers.  Default buffer size: 64 K.  Default number of buffers to send: 20 K. |
| Example Syntax for a Receiver  NTttcpr –m 1,0,10.1.2.3 –a 6 –fr | Single thread.  Bound to CPU 0.  Binding on local computer to IP 10.1.2.3.  Posting six receive overlapped buffers.  Default buffer size: 64 KB.  Default number of buffers to receive: 20 K.  Posting full-length (64 K) receive buffers. |

### Network Adapter

Make sure that you enable all offloading features.

### TCP/IP Window Size

For 1-GB adapters, the settings shown in Table 10 should provide you good throughput because NTttcp sets the default TCP window size to 64 K through a specific socket option (SO\_RCVBUF) for the connection. This provides good performance on a low-latency network. In contrast, for high-latency networks or for 10-GB adapters, NTttcp’s default TCP window size value yields less than optimal performance. In both cases, you must adjust the TCP window size to allow for the larger bandwidth delay product. You can statically set the TCP window size to a large value by using the –rb option. This option disables TCP Window Auto-Tuning, and we recommend its use only if the user fully understands the resultant change in TCP/IP behavior. By default, the TCP window size is set at a sufficient value and adjusts only under heavy load or over high-latency links.

### Receive-Side Scaling (RSS)

Windows Server 2008 supports RSS out of the box. RSS enables multiple DPCs to be scheduled and executed on concurrent processors, which improves scalability and performance for receive-intensive scenarios that have fewer networking adapters than available processors. Note that, because of hardware limitations on some adapters and to other functionality constraints, not all adapters can support concurrently processing DPCs on all processors on the server. DPCs are also not scheduled on hyperthreading processors because of an adverse effect on performance. Therefore, DPCs in RSS are scheduled only on logical and physical processors regardless of how many cores or sockets are on the server box.

## Tuning for Chariot

[Chariot](http://www.ixiacom.com/support/ixchariot/) is a networking workload generator from Ixia. It stresses the network to help predict networked application performance.

You can use the ***High\_Performance\_Throughput*** script workload of Chariot to simulate the NTttcp workload. The tuning considerations for this workload are the same as those for NTttcp.

# Performance Tuning for Terminal Server Knowledge Worker Workload

Windows Server 2008 Terminal Server capacity planning tools include automation framework and application scripting support that enable the simulation of user interaction with a Windows Terminal Server. Be aware that the following tunings apply only for a synthetic Terminal Server knowledge worker workload and are not intended as turnings for a server that is not running this workload. This workload is built with these tools to emulate common usage pattern for knowledge workers. If an updated version of the workload is released, the guide is updated accordingly.

The Terminal Server knowledge worker workload uses Microsoft Office applications and Microsoft Internet Explorer. It operates in an isolated local network that has the following infrastructure:

* Domain controller (Active Directory, Domain Name Service—DNS, and Dynamic Host Control Procedure—DHCP).
* Microsoft Exchange Server for e-mail hosting.
* Windows IIS Server for Web hosting.
* Load Generator (a test controller) for creating a distributed workload.
* A pool of Windows XP–based test systems to execute the distributed workload, with no more than 60 simulated users for each physical test system.
* Windows Terminal Server (Application Server) with Microsoft Office installed.

**Note**: The domain controller and the load generator could be combined on one physical system without degrading performance. Similarly, the IIS Server and the Exchange Server could be combined on another computer system.

Table 11 provides guidelines for achieving the best performance on the Terminal Server workload and suggestions as to where bottlenecks might exist and how to avoid them.

Table 11. Hardware Recommendations for Terminal Server Workload

| **Hardware limiting factor** | **Recommendation** |
| --- | --- |
| Processor usage | * Use 64-bit processors to expand the available virtual address space. * Use multicore systems (at least two or four sockets and dual-core or quad-core 64-bit CPUs). |
| Physical disks | * Separate the operating system files, pagefile, and user profiles (user data) to individual physical partitions. * Choose the appropriate RAID configuration. (Refer to “[Choosing the RAID Level](#_Choosing_the_RAID)” earlier in this guide.) * If applicable, set the write-through cache policy to 50% reads versus 50% writes. * If applicable, select **Enable write caching on the disk** through the Microsoft Management Console (MMC) disk management snap-in (diskmgmt.msc). * If applicable, select **Enable Advanced Performance** through the MMC disk management snap-in (diskmgmt.msc). |
| Memory (RAM) | The amount of RAM and physical memory access times affect the response times for the user interactions. On NUMA-type computer systems, make sure that the hardware configuration uses the NUMA, which is changed by using system BIOS or hardware partitioning settings. |
| Network bandwidth | Allow enough bandwidth by using network adapters that have high bandwidths such as 1‑GB Ethernet. |

## Recommended Tunings on the Server

After you have installed the operating system and added the Terminal Server role, apply the following changes:

* Navigate to **Control Panel > System > Advanced System Settings > Advanced** tab and set the following:

Navigate to **Performance Settings > Advanced > Virtual memory** and set one or more fixed-size pagefiles (**Initial Size** equal to **Maximum Size**) with a total pagefile size at least two to three times the physical RAM size to minimize paging. For servers that have hundreds of gigabytes of memory, the complete elimination of the paging file is possible. Otherwise, the paging file might be limited because of constraints in available disk space. There are no clear benefits of a paging file larger than 100 GB. Make sure that no system-managed pagefiles are in the **Virtual memory** on the Application Server.

Navigate to **Performance Settings > Visual Effects** and select the **Adjust for best performance** check box.

* Allow for the workload automation to run by opening the MMC snap-in for Group Policies (gpedit.msc) and making the following changes by navigating to **Local Computer Policy > User Configuration > Administrative Templates**:

Navigate to **Control Panel** > **Display**, and disable **Screen Saver and Password protected screen saver**.

Under **Start Menu** and **Taskbar**, enable **Force Windows Classic Start Menu**.

Navigate to **Windows Components** > **Internet Explorer**, and enable **Prevent Performance of First Run Customize settings** and select **Go directly to home page**.

Navigate to **Start > All Programs > Administrative Tools > System Configuration Tools** tab, disable User Account Control (UAC) by selecting **Disable UAC**, and then reboot the system.

* Allow for the workload automation to run by opening the registry and adding the **ProtectedModeOffForAllZones** key and set it to 1 under:

HKLM\SOFTWARE\Microsoft\Internet Explorer\Low Rights\ (REG\_DWORD)

* Minimize the effect on CPU usage when you are running many Terminal Server sessions by opening the MMC snap-in for Group Policies (gpedit.msc) and making the following changes under **Local Computer Policy > User Configuration > Administrative Templates**:

Under **Start Menu and Taskbar**, enable **Do not keep history of recently opened documents**.

Under **Start Menu and Taskbar**, enable **Remove Balloon Tips on Start Menu items**.

Under **Start Menu and Taskbar**, enable **Remove frequent program list from Start Menu**.

* Minimize the effect on the memory footprint and reduce background activity by disabling certain Microsoft Win32® services. The following are examples from command-line scripts to do this:

|  |  |
| --- | --- |
| Service name | Syntax to stop and disable service |
| Desktop Window Manager Session Manager | sc config UxSms start= disabled  sc stop UxSms |
| Windows Error Reporting service | sc config WerSvc start= disabled  sc stop WerSvc |
| Windows Update | sc config wuauserv start= disabled  sc stop wuauserv |

* Minimize background traffic by applying the following changes under **Start > All Programs > Administrative Tools > Server Manager,** and going to **Resources and Support**:

Opt out of participating in the **Customer Experience Improvement Program (CEIP)**.

Opt out of participating in **Windows Error Reporting (WER)**.

* Apply the following changes from the Terminal Services MMC snap-in (tsconfig.msc):

Set the maximum color depth to **24 bits per pixel (bpp)**.

Disable all device redirections.

Navigate to **Start > All Programs > Administrative Tools > Terminal Services > Terminal Services Configuration** and change the **Client Settings** from the **RDP-Tcp** properties as follows:

* Limit the Maximum Color Depth to 24 bpps.
* Disable redirection for all available devices such as **Drive, Windows Printer**, **LPT Port**, **COM Port**, **Clipboard**, **Audio**, **Supported Plug and Play Devices**, and **Default to main client printer.**

## Monitoring and Data Collection

The following list of performance counters is considered a base set of counters when you monitor the resource usage on the Terminal Server workload. Log the performance counters to a local, raw (blg) performance counter log. It is less expensive to collect all instances (‘\*’ wide character) and then extract particular instances while post-processing by using relog.exe.

\Cache\\*  
\IPv4\\*  
\LogicalDisk(\*)\\*  
\Memory\\*  
\Network Interface(\*)\\*  
\Paging File(\*)\\*  
\PhysicalDisk(\*)\\*  
\Print Queue(\*)\\*  
\Process(\*)\\*  
\Processor(\*)\\*  
\System\\*  
\TCPv4\\*

**Note**: If applicable, add the \IPv6\\* and \TCPv6\\* objects.

Stop unnecessary ETW loggers by running *logman.exe stop –ets <provider name>*. To view providers on the system, run *logman.exe query –ets*.

Use logman.exe to collect performance counter log data instead of using perfmon.exe, which enables logging providers and increases CPU usage.

The QIdle tool (part of Terminal Server Scaling Tools) determines whether any of the currently running scripts have failed and require an administrator to intervene. QIdle determines this by periodically checking whether any of the sessions logged on to the terminal server has been idle for longer than a specific time period. If any idle sessions exist, QIdle notifies the administrator with a beeping sound.

# Performance Tuning for SAP Sales and Distribution Two-Tier Workload

SAP AG has developed several standard application benchmarks. The Sales and Distribution (SD) workload represents one of the important classes of workloads that are used for benchmarking SAP enterprise resource planning (ERP) installations. For more information on obtaining the benchmark kit, contact [SAP](http://www.sap.com/solutions/benchmark/sd.epx).

Fine, multidimensional tuning of the operating system level, application server, database server, network, and storage is required to achieve optimal throughput and good response times as the number of concurrent SD users increases before capping out because of resource limitations.

The following are some guidelines that can benefit the two-tier setup of the SAP ERP for SD workload on Windows Server 2008.

## Operating System Tunings on the Server

* Navigate to **Control Panel > System > Advanced System Settings > Advanced** tab and set the following:

Navigate to **Performance Settings > Advanced > Virtual memory** and set one or more fixed-size pagefiles (**Initial** **Size** equal to **Maximum Size**) with a total pagefile size equal to or larger than the physical RAM size to minimize paging. For servers that have hundreds of gigabytes of memory, the entire elimination of the pagefile is possible. Otherwise, the paging file might be limited because of space constraints of available disk space. There are no clear benefits of a pagefile larger than one GB. Make sure that no system-managed pagefiles are in the **Virtual memory** on the Application Server.

Navigate to **Performance Settings > Visual Effects** and select the **Adjust for best performance** check box.

* Enable the **Lock pages in memory** user right assignment for the account that will run the SQL and SAP services.

From the Group Policy MMC snap-in (gpedit.msc), navigate to **Computer Configuration > Windows Settings > Security Settings > Local Policies > User Rights Assignment**. In the pane, double-click **Lock pages in memory** and add the accounts that have credentials to run sqlservr.exe and SAP services.

* Disable User Account Control.

Navigate to **Start > All Programs > Administrative Tools > System Configuration > Tools** tab, start **Disable UAC**, and then reboot the system.

## Tunings on the Database Server

When the database server is SQL Server® 2005, consider setting the following SQL Server configuration options with *sp\_configure*. For detailed information on the *sp\_configure* stored procedure, refer to “[Setting Server Configuration Options](http://go.microsoft.com/fwlink/?LinkId=98291).”

* Apply CPU core affinity for the SQL Server 2005 process: Set **affinity mask** and **affinity I/O mask** to partition SQL process on specific cores. If required, use the **affinity64 mask** and **affinity64 I/O mask** to set the affinity on more than 32 cores.
* On NUMA class hardware, do the following:

To further subdivide the CPUs in a hardware NUMA node to more CPU nodes (known as Soft-NUMA), refer to “[How to: Configure SQL Server to Use Soft-NUMA](http://go.microsoft.com/fwlink/?LinkId=98292).”

To set TCP/IP connection affinity, refer to “[How to: Map TCP/IP Ports to NUMA Nodes](http://go.microsoft.com/fwlink/?LinkId=98293).”

* Set a fixed amount of memory that the SQL Server process will use. For example, set the **max server memory** and **min server memory** equal and large enough to satisfy the workload (2500 MB is a good starting value).
* Change the **network packet size** to 8 KB for better page alignment in SQL environments.
* Set the **recovery interval** to 32767, to offset the SQL Server checkpoints while it is running the workload.
* On a two-tier ERP SAP setup, consider enabling and using only the **Named Pipes** protocol and disabling the rest of the available protocols from the **SQL Server Configuration Manager** for the local SQL connections.

## Tunings on the SAP Application Server

* The ratio between the number of Dialog Instances (D) versus Update (U) instances in the SAP ERP installation might vary, but usually a ratio of 1:1U or 2D:1U is a good start for the SD workload.
* Use the processor affinity capabilities in the SAP’s instance profiles to partition each worker process to a subset of the available CPU cores and therefore achieve better CPU and memory locality.
* Use the FLAT memory model that SAP AG released on November 23, 2006, with the SAP Note No. 1002587 “Flat Memory Model on Windows” for SAP kernel 7.00 Patch Level 87.

## Monitoring and Data Collection

The following list of performance counters is considered a base set of counters when you monitor the resource usage of the Application Server while you are running the two-tier SAP ERP SD workload. Log the performance counters to a local, raw (blg) performance counter log. It is less expensive to collect all instances (‘\*’ wide character) and then extract particular instances while post-processing by using relog.exe:

\Cache\\*  
\IPv4\\*  
\LogicalDisk(\*)\\*  
\Memory\\*  
\Network Interface(\*)\\*  
\Paging File(\*)\\*  
\PhysicalDisk(\*)\\*  
\Process(\*)\\*  
\Processor(\*)\\*  
\System\\*  
\TCPv4\\*

**Note**: If applicable, add the \IPv6\\* and \TCPv6\\* objects.

# Resources

Web Sites:

Windows Server 2008

<http://www.microsoft.com/windowsserver2008>

Windows Server Performance Team Blog

<http://blogs.technet.com/winserverperformance/>

SAP Global

<http://www.sap.com/solutions/benchmark/sd.epx>

Transaction Processing Performance Council

<http://www.tpc.org>

Documents:

Scalable Networking: Eliminating the Receive Processing Bottleneck—Introducing RSS

<http://download.microsoft.com/download/5/D/6/5D6EAF2B-7DDF-476B-93DC-7CF0072878E6/NDIS_RSS.doc>

Disk Subsystem Performance Analysis for Windows

<http://www.microsoft.com/whdc/device/storage/subsys_perf.mspx>

10 Tips for Writing High-Performance Web Applications

<http://go.microsoft.com/fwlink/?LinkId=98290>

Performance Tuning Guidelines for Microsoft Services for Network File System

<http://technet.microsoft.com/en-us/library/bb463205.aspx>

Active Directory Performance for 64-bit Versions of Windows Server 2003

<http://www.microsoft.com/downloads/details.aspx?FamilyID=52e7c3bd-570a-475c-96e0-316dc821e3e7>

How to configure Active Directory diagnostic event logging in Windows Server 2003 and in Windows 2000 Server

<http://support.microsoft.com/kb/314980>

Setting Server Configuration Options

<http://go.microsoft.com/fwlink/?LinkId=98291>

How to: Configure SQL Server to Use Soft-NUMA

<http://go.microsoft.com/fwlink/?LinkId=98292>

How to: Map TCP/IP Ports to NUMA Nodes

<http://go.microsoft.com/fwlink/?LinkId=98293>

**SAP with Microsoft SQL Server 2005:**

**Best Practices for High Availability, Maximum Performance, and Scalability**

<http://download.microsoft.com/download/d/9/4/d948f981-926e-40fa-a026-5bfcf076d9b9/SAP_SQL2005_Best%20Practices.doc>