Contents

Performance Tuning Guidelines for Windows Server 2022

12/16/2022 • 2 minutes to read • Edit Online

When you run a server system in your organization, you might have business needs not met using default server settings. For example, you might need the lowest possible energy consumption, or the lowest possible latency, or the maximum possible throughput on your server. This guide provides a set of guidelines that you can use to tune the server settings in Windows Server 2022 and obtain incremental performance or energy efficiency gains, especially when the nature of the workload varies little over time.

It is important that your tuning changes consider the hardware, the workload, the power budgets, and the performance goals of your server. This guide describes each setting and its potential effect to help you make an informed decision about its relevance to your system, workload, performance, and energy usage goals.

WARNING

Registry settings and tuning parameters changed significantly between versions of Windows Server. Be sure to use the latest tuning guidelines to avoid unexpected results.

In this guide

This guide organizes performance and tuning guidance for Windows Server 2022 across three tuning categories:

SERVER HARDWARE	SERVER ROLE	SERVER SUBSYSTEM
Hardware performance considerations	Active Directory Servers	Cache and memory management
Hardware power considerations	File Servers	Networking subsystem
	Hyper-V Servers	Storage Spaces Direct
	Remote Desktop Services	Software Defined Networking (SDN)
	Web Servers	
	Windows Server Containers	

Changes in this version

Sections added

- Software Defined Networking, including HNV and SLB gateway configuration guidance
- Storage Spaces Direct
- HTTP1.1 and HTTP2
- Windows Server Containers

Sections changed

- Updates to Active Directory guidance section
- Updates to File Server guidance section
- Updates to Web Server guidance section
- Updates to Hardware Power guidance section
- Updates to PowerShell tuning guidance section
- Significant updates to the Hyper-V guidance section
- *Performance Tuning for Workloads removed*, pointers to relevant resources added to Additional Tuning Resources article
- *Removal of dedicated storage sections*, in favor of new Storage Spaces Direct section and canonical Technet content
- *Removal of dedicated networking section*, in favor of canonical Technet content

Server Hardware Performance Considerations

12/16/2022 • 6 minutes to read • Edit Online

The following section lists important items that you should consider when you choose server hardware. Following these guidelines can help remove performance bottlenecks that might impede the server's performance.

Processor Recommendations

Choose 64-bit processors for servers. 64-bit processors have significantly more address space, and are required for Windows Server 2022. No 32-bit editions of the operating system will be provided, but 32-bit applications will run on the 64-bit Windows Server 2022 operating system.

To increase the computing resources in a server, you can use a processor with higher-frequency cores, or you can increase the number of processor cores. If CPU is the limiting resource in the system, a core with 2x frequency typically provides a greater performance improvement than two cores with 1x frequency.

Multiple cores are not expected to provide a perfect linear scaling, and the scaling factor can be even less if hyper-threading is enabled because hyper-threading relies on sharing resources of the same physical core.

> **IMPORTANT**
> Match and scale the memory and I/O subsystem with the CPU performance, and vice versa.

Do not compare CPU frequencies across manufacturers and generations of processors because the comparison can be a misleading indicator of speed.

For Hyper-V, make sure that the processor supports SLAT (Second Level Address Translation). It is implemented as Extended Page Tables (EPT) by Intel and Nested Page Tables (NPT) by AMD. You can verify this feature is present by using SystemInfo.exe on your server.

Cache Recommendations

Choose large L2 or L3 processor caches. On newer architectures, such as Haswell or Skylake, there is a unified Last Level Cache (LLC) or an L4. The larger caches generally provide better performance, and they often play a bigger role than raw CPU frequency.

Memory (RAM) and Paging Storage Recommendations

> **NOTE**
> Some systems may exhibit reduced storage performance when running a new install of Windows Server 2022 versus Windows Server 2012 R2. A number of changes were made during the development of Windows Server 2022 to improve security and reliability of the platform. Some of those changes, such as enabling Windows Defender by default, result in longer I/O paths that can reduce I/O performance in specific workloads and patterns. Microsoft does not recommend disabling Windows Defender as it is an important layer of protection for your systems.

Increase the RAM to match your memory needs. When your computer runs low on memory and it needs more immediately, Windows uses hard disk space to supplement system RAM through a procedure called paging. Too much paging degrades the overall system performance. You can optimize paging by using the following

guidelines for page file placement:

- Isolate the page file on its own storage device, or at least make sure it doesn't share the same storage devices as other frequently accessed files. For example, place the page file and operating system files on separate physical disk drives.
- Place the page file on a drive that is fault-tolerant. If a non-fault-tolerant disk fails, a system crash is likely to occur. If you place the page file on a fault-tolerant drive, remember that fault-tolerant systems are often slower to write data because they write data to multiple locations.
- Use multiple disks or a disk array if you need additional disk bandwidth for paging. Do not place multiple page files on different partitions of the same physical disk drive.

Peripheral Bus Recommendations

In Windows Server 2022, the primary storage and network interfaces should be PCI Express (PCIe) so servers with PCIe buses are recommended. To avoid bus speed limitations, use PCIe x8 and higher slots for 10+ GB Ethernet adapters.

Disk Recommendations

Choose disks with higher rotational speeds to reduce random request service times (~2 ms on average when you compare 7,200- and 15,000-RPM drives) and to increase sequential request bandwidth. However, there are cost, power, and other considerations associated with disks that have high rotational speeds.

2.5-inch enterprise-class disks can service a significantly larger number of random requests per second compared to equivalent 3.5-inch drives.

Store frequently accessed data, especially sequentially accessed data, near the beginning of a disk because this roughly corresponds to the outermost (fastest) tracks.

Consolidating small drives into fewer high-capacity drives can 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).

The use of SSD and high speed flash disks is useful for read mostly disks with high I/O rates or latency sensitive I/O. Boot disks are good candidates for the use of SSD or high speed flash disks as they can improve boot times significantly.

NVMe SSDs offer superior performance with greater command queue depths, more efficient interrupt processing, and greater efficiency for 4KB commands. This particularly benefits scenarios that requires heavy simultaneous I/O.

Network and Storage Adapter Recommendations

The following section lists the recommended characteristics for network and storage adapters for high-performance servers. These settings can help prevent your networking or storage hardware from being a bottleneck when they are under heavy load.

Certified adapter usage

Use an adapter that has passed the Windows Hardware 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 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 a limited number of 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 adapters 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 you can attach more disks 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 PCIe slot that provides enough bandwidth.

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.

Receive Side Scaling (RSS) support

RSS enables packet receive-processing to scale with the number of available computer processors. This is particularly important with 10 GB Ethernet and faster.

Offload capability and other advanced features such as message-signaled interrupt (MSI)-X

Offload-capable adapters offer CPU savings that yield improved performance.

Dynamic interrupt and deferred procedure call (DPC) redirection

In Windows Server 2022, Numa I/O enables PCIe storage adapters to dynamically redirect interrupts and DPCs and can help any multiprocessor system by improving workload partitioning, cache hit rates, and on-board hardware interconnect usage for I/O-intensive workloads.

See Also

- Server Hardware Power Considerations
- Overview about power and performance tuning for the Windows Server
- Processor Power Management (PPM) tuning for the Windows Server balanced power plan

Server Hardware Performance Considerations

12/16/2022 • 6 minutes to read • Edit Online

The following section lists important items that you should consider when you choose server hardware. Following these guidelines can help remove performance bottlenecks that might impede the server's performance.

Processor Recommendations

Choose 64-bit processors for servers. 64-bit processors have significantly more address space, and are required for Windows Server 2022. No 32-bit editions of the operating system will be provided, but 32-bit applications will run on the 64-bit Windows Server 2022 operating system.

To increase the computing resources in a server, you can use a processor with higher-frequency cores, or you can increase the number of processor cores. If CPU is the limiting resource in the system, a core with 2x frequency typically provides a greater performance improvement than two cores with 1x frequency.

Multiple cores are not expected to provide a perfect linear scaling, and the scaling factor can be even less if hyper-threading is enabled because hyper-threading relies on sharing resources of the same physical core.

> **IMPORTANT**
> Match and scale the memory and I/O subsystem with the CPU performance, and vice versa.

Do not compare CPU frequencies across manufacturers and generations of processors because the comparison can be a misleading indicator of speed.

For Hyper-V, make sure that the processor supports SLAT (Second Level Address Translation). It is implemented as Extended Page Tables (EPT) by Intel and Nested Page Tables (NPT) by AMD. You can verify this feature is present by using SystemInfo.exe on your server.

Cache Recommendations

Choose large L2 or L3 processor caches. On newer architectures, such as Haswell or Skylake, there is a unified Last Level Cache (LLC) or an L4. The larger caches generally provide better performance, and they often play a bigger role than raw CPU frequency.

Memory (RAM) and Paging Storage Recommendations

> **NOTE**
> Some systems may exhibit reduced storage performance when running a new install of Windows Server 2022 versus Windows Server 2012 R2. A number of changes were made during the development of Windows Server 2022 to improve security and reliability of the platform. Some of those changes, such as enabling Windows Defender by default, result in longer I/O paths that can reduce I/O performance in specific workloads and patterns. Microsoft does not recommend disabling Windows Defender as it is an important layer of protection for your systems.

Increase the RAM to match your memory needs. When your computer runs low on memory and it needs more immediately, Windows uses hard disk space to supplement system RAM through a procedure called paging. Too much paging degrades the overall system performance. You can optimize paging by using the following

guidelines for page file placement:

- Isolate the page file on its own storage device, or at least make sure it doesn't share the same storage devices as other frequently accessed files. For example, place the page file and operating system files on separate physical disk drives.
- Place the page file on a drive that is fault-tolerant. If a non-fault-tolerant disk fails, a system crash is likely to occur. If you place the page file on a fault-tolerant drive, remember that fault-tolerant systems are often slower to write data because they write data to multiple locations.
- Use multiple disks or a disk array if you need additional disk bandwidth for paging. Do not place multiple page files on different partitions of the same physical disk drive.

Peripheral Bus Recommendations

In Windows Server 2022, the primary storage and network interfaces should be PCI Express (PCIe) so servers with PCIe buses are recommended. To avoid bus speed limitations, use PCIe x8 and higher slots for 10+ GB Ethernet adapters.

Disk Recommendations

Choose disks with higher rotational speeds to reduce random request service times (~2 ms on average when you compare 7,200- and 15,000-RPM drives) and to increase sequential request bandwidth. However, there are cost, power, and other considerations associated with disks that have high rotational speeds.

2.5-inch enterprise-class disks can service a significantly larger number of random requests per second compared to equivalent 3.5-inch drives.

Store frequently accessed data, especially sequentially accessed data, near the beginning of a disk because this roughly corresponds to the outermost (fastest) tracks.

Consolidating small drives into fewer high-capacity drives can 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).

The use of SSD and high speed flash disks is useful for read mostly disks with high I/O rates or latency sensitive I/O. Boot disks are good candidates for the use of SSD or high speed flash disks as they can improve boot times significantly.

NVMe SSDs offer superior performance with greater command queue depths, more efficient interrupt processing, and greater efficiency for 4KB commands. This particularly benefits scenarios that requires heavy simultaneous I/O.

Network and Storage Adapter Recommendations

The following section lists the recommended characteristics for network and storage adapters for high-performance servers. These settings can help prevent your networking or storage hardware from being a bottleneck when they are under heavy load.

Certified adapter usage

Use an adapter that has passed the Windows Hardware 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 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 a limited number of 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 adapters 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 you can attach more disks 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 PCIe slot that provides enough bandwidth.

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.

Receive Side Scaling (RSS) support

RSS enables packet receive-processing to scale with the number of available computer processors. This is particularly important with 10 GB Ethernet and faster.

Offload capability and other advanced features such as message-signaled interrupt (MSI)-X

Offload-capable adapters offer CPU savings that yield improved performance.

Dynamic interrupt and deferred procedure call (DPC) redirection

In Windows Server 2022, Numa I/O enables PCIe storage adapters to dynamically redirect interrupts and DPCs and can help any multiprocessor system by improving workload partitioning, cache hit rates, and on-board hardware interconnect usage for I/O-intensive workloads.

See Also

- Server Hardware Power Considerations
- Overview about power and performance tuning for the Windows Server
- Processor Power Management (PPM) tuning for the Windows Server balanced power plan

Server Hardware Power Considerations

12/16/2022 • 2 minutes to read • Edit Online

It is important to recognize the increasing importance of energy efficiency in enterprise and data center environments. High performance and low-energy usage are often conflicting goals, but by carefully selecting server components, you can achieve the correct balance between them. The following sections lists guidelines for power characteristics and capabilities of server hardware components.

Processor Recommendations

Frequency, operating voltage, cache size, and process technology affect the energy consumption of processors. Processors have a thermal design point (TDP) rating that gives a basic indication of energy consumption relative to other models.

In general, opt for the lowest TDP processor that will meet your performance goals. Also, newer generations of processors are generally more energy efficient, and they may expose more power states for the Windows power management algorithms, which enables better power management at all levels of performance. Or they may use some of the new "cooperative" power management techniques that Microsoft has developed in partnership with hardware manufacturers.

For more info on cooperative power management techniques, see the section named Collaborative Processor Performance Control in the Advanced Configuration and Power Interface Specification.

Memory Recommendations

Memory accounts for an increasing fraction of the total system power. Many factors affect the energy consumption of a memory DIMM, such as memory technology, error correction code (ECC), bus frequency, capacity, density, and number of ranks. Therefore, it is best to compare expected power ratings before purchasing large quantities of memory.

Low-power memory is now available, but you must consider the performance and cost trade-offs. If your server will be paging, you should also factor in the energy cost of the paging disks.

Disks Recommendations

Higher RPM means increased energy consumption. SSD drives are more power efficient than rotational drives. Also, 2.5-inch drives generally require less power than 3.5-inch drives.

Network and Storage Adapter Recommendations

Some adapters decrease energy consumption during idle periods. This is an important consideration for 10 Gb networking adapters and high-bandwidth (4-8 Gb) storage links. Such devices can consume significant amounts of energy.

Power Supply Recommendations

Improving power supply efficiency is a great way to reduce energy consumption without affecting performance. High-efficiency power supplies can save many kilowatt-hours per year, per server.

Fan Recommendations

Fans, like power supplies, are an area where you can reduce energy consumption without affecting system performance. Variable-speed fans can reduce RPM as the system load decreases, eliminating otherwise unnecessary energy consumption.

USB devices Recommendations

Windows Server 2016 enables selective suspend for USB devices by default. However, a poorly written device driver can still disrupt system energy efficiency by a sizeable margin. To avoid potential issues, disconnect USB devices, disable them in the BIOS, or choose servers that do not require USB devices.

Remotely-managed Power Strip Recommendations

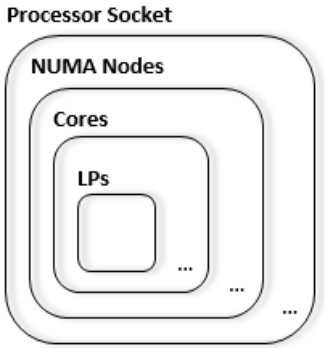
Power strips are not an integral part of server hardware, but they can make a large difference in the data center. Measurements show that volume servers that are plugged in, but have been ostensibly powered off, may still require up to 30 watts of power.

To avoid wasting electricity, you can deploy a remotely managed power strip for each rack of servers to programmatically disconnect power from specific servers.

Processor terminology

The processor terminology used throughout this topic reflects the hierarchy of components available in the following figure. Terms used from largest to smallest granularity of components are the following:

- Processor socket
- NUMA node
- Core
- Logical processor



Additional References

- Server Hardware Performance Considerations
- Overview about power and performance tuning for the Windows Server
- Processor Power Management (PPM) tuning for the Windows Server balanced power plan

Power and performance tuning

12/16/2022 • 17 minutes to read • Edit Online

Energy efficiency is increasingly important in enterprise and data center environments, and it adds another set of tradeoffs to the mix of configuration options. When managing servers, it's important to ensure that they are running as efficiently as possible while meeting the performance needs of their workloads. Windows Server is optimized for excellent energy efficiency with minimum performance impact across a wide range of customer workloads. Processor Power Management (PPM) Tuning for the Windows Server Balanced Power Plan describes the workloads used for tuning the default parameters in multiple Windows Server versions, and provides suggestions for customized tunings.

This section expands on energy-efficiency tradeoffs to help you make informed decisions if you need to adjust the default power settings on your server. However, the majority of server hardware and workloads should not require administrator power tuning when running Windows Server.

Choosing the tuning metrics

When you tune your server for energy savings, you must also consider performance. Tuning affects performance and power, sometimes in disproportionate amounts. For each possible adjustment, consider your power budget and performance goals to determine whether the trade-off is acceptable.

Windows Server default parameter tuning uses Energy Efficiency as a key metric to balance power and performance. Energy efficiency is the ratio of work that is done to the average power that is required during a specified amount of time.

$$Energy\ Efficiency = \frac{Rate\ of\ Work\ Done}{Average\ Watts\ Of\ Power\ Required}$$

You can use this metric to set practical goals that respect the tradeoff between power and performance. In contrast, a goal of 10 percent energy savings across the data center fails to capture the corresponding effects on performance and vice versa.

Similarly, if you tune your server to increase performance by 5 percent, and that results in 10 percent higher energy consumption, the total result might or might not be acceptable for your business goals. The energy efficiency metric allows for more informed decision making than power or performance metrics alone.

Measuring system energy consumption

You should establish a baseline power measurement before you tune your server for energy efficiency.

If your server has the necessary support, you can use the power metering and budgeting features in Windows Server 2016 to view system-level energy consumption by using Performance Monitor.

One way to determine whether your server has support for metering and budgeting is to review the Windows Server Catalog. If your server model qualifies for the new Enhanced Power Management qualification in the Windows Hardware Certification Program, it is guaranteed to support the metering and budgeting functionality.

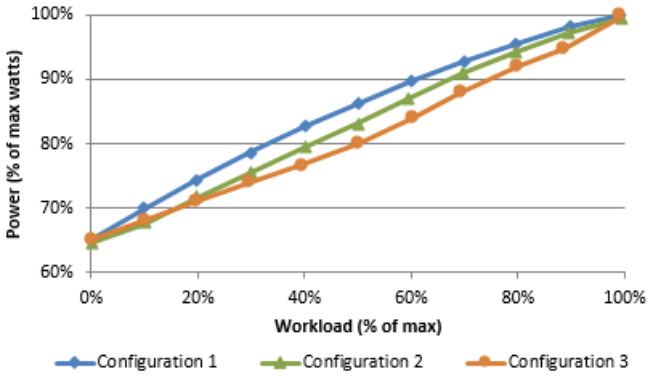
Another way to check for metering support is to manually look for the counters in Performance Monitor. Open Performance Monitor, select **Add Counters**, and then locate the **Power Meter** counter group.

If named instances of power meters appear in the box labeled **Instances of Selected Object**, your platform supports metering. The **Power** counter that shows power in watts appears in the selected counter group. The exact derivation of the power data value is not specified. For example, it could be an instantaneous power draw

or an average power draw over some time interval.

If your server platform does not support metering, you can use a physical metering device connected to the power supply input to measure system power draw or energy consumption.

To establish a baseline, you should measure the average power required at various system load points, from idle to 100 percent (maximum throughput) to generate a load line. The following figure shows load lines for three sample configurations:



You can use load lines to evaluate and compare the performance and energy consumption of configurations at all load points. In this particular example, it is easy to see what the best configuration is. However, there can easily be scenarios where one configuration works best for heavy workloads and one works best for light workloads.

You need to thoroughly understand your workload requirements to choose an optimal configuration. Don't assume that when you find a good configuration, it will always remain optimal. You should measure system utilization and energy consumption on a regular basis and after changes in workloads, workload levels, or server hardware.

Diagnosing energy efficiency issues

PowerCfg.exe supports a command-line option that you can use to analyze the idle energy efficiency of your server. When you run PowerCfg.exe with the **/energy** option, the tool performs a 60-second test to detect potential energy efficiency issues. The tool generates a simple HTML report in the current directory.

> **IMPORTANT**
> To ensure an accurate analysis, make sure that all local apps are closed before you run **PowerCfg.exe**.

Shortened timer tick rates, drivers that lack power management support, and excessive CPU utilization are a few of the behavioral issues that are detected by the **powercfg /energy** command. This tool provides a simple way to identify and fix power management issues, potentially resulting in significant cost savings in a large datacenter.

For more info about PowerCfg.exe, see Powercfg command-line options.

Using power plans in Windows Server

Windows Server 2016 has three built-in power plans designed to meet different sets of business needs. These plans provide a simple way for you to customize a server to meet power or performance goals. The following table describes the plans, lists the common scenarios in which to use each plan, and gives some implementation details for each plan.

PLAN	DESCRIPTION	COMMON APPLICABLE SCENARIOS	IMPLEMENTATION HIGHLIGHTS
Balanced (recommended)	Default setting. Targets good energy efficiency with minimal performance impact.	General computing	Matches capacity to demand. Energy-saving features balance power and performance.
High Performance	Increases performance at the cost of high energy consumption. Power and thermal limitations, operating expenses, and reliability considerations apply.	Low latency apps and app code that is sensitive to processor performance changes	Processors are always locked at the highest performance state (including "turbo" frequencies). All cores are unparked. Thermal output may be significant.
Power Saver	Limits performance to save energy and reduce operating cost. Not recommended without thorough testing to make sure performance is adequate.	Deployments with limited power budgets and thermal constraints	Caps processor frequency at a percentage of maximum (if supported), and enables other energy-saving features.

These power plans exist in Windows for alternating current (AC) and direct current (DC) powered systems, but we will assume that servers are always using an AC power source.

For more info on power plans and power policy configurations, see Powercfg command-line options.

> **NOTE**
> Some server manufactures have their own power management options available through the BIOS settings. If the operating system does not have control over the power management, changing the power plans in Windows will not affect system power and performance.

Tuning processor power management parameters

Each power plan represents a combination of numerous underlying power management parameters. The built-in plans are three collections of recommended settings that cover a wide variety of workloads and scenarios. However, we recognize that these plans will not meet every customer's needs.

The following sections describe ways to tune some specific processor power management parameters to meet goals not addressed by the three built-in plans. If you need to understand a wider array of power parameters, see Powercfg command-line options.

Intel Hardware Controlled P-states (HWP)

Starting from Intel Broadwell processors running WS2016, Windows PPM uses Intel's Hardware Controlled P-states (HWP). HWP is a new capability for a cooperative hardware and software performance control. When HWP is enabled, CPU monitors activity and scalability, and selects frequency at hardware time scale. OS is no longer required to monitor activity and select frequency at regular intervals. Switching to HWP has several benefits:

- Quickly respond to bursty workloads. Windows PPM check interval is set as 30ms as default and can be reduced as minimal as 15ms. However, HWP can adjust frequency at quick as every 1ms.
- CPU has better knowledge of the hardware power efficiency of each P-state. It can make a better choice of

processor frequency to achieve the best power efficiency.
- CPU can take other hardware usage, e.g., memory, GPU, etc., into account to achieve best power efficiency under certain TDP (Thermal Design Power).

Windows can still set the minimum and maximum processor states to limit the range of frequencies that the processors can execute. It can also set the following Processor energy performance preference policy (EPP) parameter to indicate HWP to favor power or performance.

- **Processor energy performance preference policy** to set balance between power and performance. Lower value favors performance, and higher value favors power. The value can be between 0 and 100. The default value 50 that is to balance power and performance.

The following commands decrease the EPP value to 0 on current power plan to totally favor performance over power:

```
Powercfg -setacvalueindex scheme_current sub_processor PERFEPP 0
Powercfg -setactive scheme_current
```

Minimum and maximum processor performance state

Processors change between performance states (P-states) very quickly to match supply to demand, delivering performance where necessary and saving energy when possible. If your server has specific high-performance or minimum-power-consumption requirements, you might consider configuring the **Minimum Processor Performance State** parameter or the **Maximum Processor Performance State** parameter.

The values for the **Minimum Processor Performance State** and **Maximum Processor Performance State** parameters are expressed as a percentage of maximum processor frequency, with a value in the range 0 – 100.

If your server requires ultra-low latency, invariant CPU frequency (e.g., for repeatable testing), or the highest performance levels, you might not want the processors switching to lower-performance states. For such a server, you can cap the minimum processor performance state at 100 percent by using the following commands:

```
Powercfg -setacvalueindex scheme_current sub_processor PROCTHROTTLEMIN 100
Powercfg -setactive scheme_current
```

If your server requires lower energy consumption, you might want to cap the processor performance state at a percentage of maximum. For example, you can restrict the processor to 75 percent of its maximum frequency by using the following commands:

```
Powercfg -setacvalueindex scheme_current sub_processor PROCTHROTTLEMAX 75
Powercfg -setactive scheme_current
```

> **NOTE**
> Capping processor performance at a percentage of maximum requires processor support. Check the processor documentation to determine whether such support exists, or view the Performance Monitor counter **% of maximum frequency** in the **Processor** group to see if any frequency caps were applied.

Processor responsiveness override

The CPU utilization-based power management algorithms typically uses a average CPU utilization within a time check window to determine if frequency needs to increase or decrease. That might hurt the latency of disk I/O or

network heavy workloads. A logical processor could be idle while waiting for disk I/O completion or network packets, which makes the overall CPU utilization low. As a result, power management will choose a low frequency for this processor. This issue exists on HWP-based power management as well. The DPCs and threads handling the IO completion or network packets are in the critical path and should not run at low speed. To resolve this issue, Windows PPM takes the number of DPCs into account. When the DPC count is above certain threshold in the past monitoring window, PPM will enter an IO responsiveness period and raises the frequency floor to a higher level. The frequency floor will be reset when the DPC count is low enough for some time. The behavior can be tuned by the following parameters.

PARAMETER	DESCRIPTION	DEFAULT VALUE	MIN VALUE	MAX VALUE
Processor responsiveness override enable threshold	Count of DPCs within a perf check above which processor responsiveness overrides should be enabled	10	0	N/A
Processor responsiveness override disable threshold	Count of DPCs within a perf check below which processor responsiveness overrides should be disabled	5	0	N/A
Processor responsiveness override enable time	Count of consecutive perf checks which must meet the enable threshold before processor responsiveness overrides are enabled	1	1	100
Processor responsiveness override disable time	Count of consecutive perf checks which must meet the disable threshold before processor responsiveness overrides are disabled	3	1	100
Processor responsiveness override performance floor	Minimum allowed processor performance when processor responsiveness overrides are enabled	100	0	100
Processor responsiveness override energy performance preference ceiling	Maximum energy performance preference policy value when processor responsiveness overrides are enabled	100	0	100

For example, if your server workload is not sensitive to the latency and wants to loose the responsiveness override to favor power, you can increase the Processor responsiveness override enable threshold and Processor responsiveness override enable time, decrease the Processor responsiveness override disable

threshold and Processor responsiveness override disable time. Then the system will be hard to enter responsiveness override state. The default value of Processor responsiveness override performance floor is set as 100 so that the responsiveness override period will run at maximum frequency. You can also decrease the processor performance floor and reduce the Processor responsiveness override energy performance preference ceiling to let HWP to adjust the frequency. The following are the sample commands to set the parameters for current active power plan.

```
Powercfg -setacvalueindex scheme_current sub_processor RESPENABLETHRESHOLD 100
Powercfg -setacvalueindex scheme_current sub_processor RESPDISABLETHRESHOLD 1
Powercfg -setacvalueindex scheme_current sub_processor RESPENABLETIME 10
Powercfg -setacvalueindex scheme_current sub_processor RESPDISABLETIME 1
Powercfg -setacvalueindex scheme_current sub_processor RESPPERFFLOOR 5
Powercfg -setacvalueindex scheme_current sub_processor RESPEPPCEILING 50
Powercfg -setactive scheme_current
```

Processor performance boost mode

This parameter tuning only applies to Non-HWP systems.

Intel Turbo Boost and AMD Turbo CORE technologies are features that allow processors to achieve additional performance when it is most useful (that is, at high system loads). However, this feature increases CPU core energy consumption, so Windows Server 2016 configures Turbo technologies based on the power policy that is in use and the specific processor implementation.

Turbo is enabled for High Performance power plans on all Intel and AMD processors and it is disabled for Power Saver power plans. For Balanced power plans on systems that rely on traditional P-state-based frequency management, Turbo is enabled by default only if the platform supports the EPB register.

> **NOTE**
> The EPB register is only supported in Intel Westmere and later processors.

For Intel Nehalem and AMD processors, Turbo is disabled by default on P-state-based platforms. However, if a system supports Collaborative Processor Performance Control (CPPC), which is a new alternative mode of performance communication between the operating system and the hardware (defined in ACPI 5.0), Turbo may be engaged if the Windows operating system dynamically requests the hardware to deliver the highest possible performance levels.

To enable or disable the Turbo Boost feature, the Processor Performance Boost Mode parameter must be configured by the administrator or by the default parameter settings for the chosen power plan. Processor Performance Boost Mode has five allowable values, as shown in Table 5.

For P-state-based control, the choices are Disabled, Enabled (Turbo is available to the hardware whenever nominal performance is requested), and Efficient (Turbo is available only if the EPB register is implemented).

For CPPC-based control, the choices are Disabled, Efficient Enabled (Windows specifies the exact amount of Turbo to provide), and Aggressive (Windows asks for "maximum performance" to enable Turbo).

In Windows Server 2016, the default value for Boost Mode is 3.

NAME	P-STATE-BASED BEHAVIOR	CPPC BEHAVIOR
0 (Disabled)	Disabled	Disabled
1 (Enabled)	Enabled	Efficient Enabled

NAME	P-STATE-BASED BEHAVIOR	CPPC BEHAVIOR
2 (Aggressive)	Enabled	Aggressive
3 (Efficient Enabled)	Efficient	Efficient Enabled
4 (Efficient Aggressive)	Efficient	Aggressive

The following commands enable Processor Performance Boost Mode on the current power plan (specify the policy by using a GUID alias):

```
Powercfg -setacvalueindex scheme_current sub_processor PERFBOOSTMODE 1
Powercfg -setactive scheme_current
```

> **IMPORTANT**
> You must run the **powercfg -setactive** command to enable the new settings. You do not need to reboot the server.

To set this value for power plans other than the currently selected plan, you can use aliases such as SCHEME_MAX (Power Saver), SCHEME_MIN (High Performance), and SCHEME_BALANCED (Balanced) in place of SCHEME_CURRENT. Replace "scheme current" in the powercfg -setactive commands previously shown with the desired alias to enable that power plan.

For example, to adjust the Boost Mode in the Power Saver plan and make that Power Saver is the current plan, run the following commands:

```
Powercfg -setacvalueindex scheme_max sub_processor PERFBOOSTMODE 1
Powercfg -setactive scheme_max
```

Processor performance increase and decrease of thresholds and policies

This parameter tuning only applies to Non-HWP systems.

The speed at which a processor performance state increases or decreases is controlled by multiple parameters. The following four parameters have the most visible impact:

- **Processor Performance Increase Threshold** defines the utilization value above which a processor's performance state will increase. Larger values slow the rate of increase for the performance state in response to increased activities.
- **Processor Performance Decrease Threshold** defines the utilization value below which a processor's performance state will decrease. Larger values increase the rate of decrease for the performance state during idle periods.
- **Processor Performance Increase Policy and Processor Performance Decrease** Policy determine which performance state should be set when a change happens. "Single" policy means it chooses the next state. "Rocket" means the maximum or minimal power performance state. "Ideal" tries to find a balance between power and performance.

For example, if your server requires ultra-low latency while still wanting to benefit from low power during idle periods, you could quicken the performance state increase for any increase in load and slow the decrease when load goes down. The following commands set the increase policy to "Rocket" for a faster state increase, and set

the decrease policy to "Single". The increase and decrease thresholds are set to 10 and 8 respectively.

```
Powercfg.exe -setacvalueindex scheme_current sub_processor PERFINCPOL 2
Powercfg.exe -setacvalueindex scheme_current sub_processor PERFDECPOL 1
Powercfg.exe -setacvalueindex scheme_current sub_processor PERFINCTHRESHOLD 10
Powercfg.exe -setacvalueindex scheme_current sub_processor PERFDECTHRESHOLD 8
Powercfg.exe /setactive scheme_current
```

Processor performance core parking maximum and minimum cores

Core parking is a feature that was introduced in Windows Server 2008 R2. The processor power management (PPM) engine and the scheduler work together to dynamically adjust the number of cores that are available to run threads. The PPM engine chooses a minimum number of cores for the threads that will be scheduled.

Cores that are parked generally do not have any threads scheduled, and they will drop into very low power states when they are not processing interrupts, DPCs, or other strictly affinitized work. The remaining cores are responsible for the remainder of the workload. Core parking can potentially increase energy efficiency during lower usage.

For most servers, the default core-parking behavior provides a reasonable balance of throughput and energy efficiency. On processors where core parking may not show as much benefit on generic workloads, it can be disabled by default.

If your server has specific core parking requirements, you can control the number of cores that are available to park by using the **Processor Performance Core Parking Maximum Cores** parameter or the **Processor Performance Core Parking Minimum Cores** parameter in Windows Server 2016.

One scenario that core parking isn't always optimal for is when there are one or more active threads affinitized to a non-trivial subset of CPUs in a NUMA node (that is, more than 1 CPU, but less than the entire set of CPUs on the node). When the core parking algorithm is picking cores to unpark (assuming an increase in workload intensity occurs), it may not always pick the cores within the active affinitized subset (or subsets) to unpark, and thus may end up unparking cores that won't actually be utilized.

The values for these parameters are percentages in the range 0 – 100. The **Processor Performance Core Parking Maximum Cores** parameter controls the maximum percentage of cores that can be unparked (available to run threads) at any time, while the **Processor Performance Core Parking Minimum Cores** parameter controls the minimum percentage of cores that can be unparked. To turn off core parking, set the **Processor Performance Core Parking Minimum Cores** parameter to 100 percent by using the following commands:

```
Powercfg -setacvalueindex scheme_current sub_processor CPMINCORES 100
Powercfg -setactive scheme_current
```

To reduce the number of schedulable cores to 50 percent of the maximum count, set the **Processor Performance Core Parking Maximum Cores** parameter to 50 as follows:

```
Powercfg -setacvalueindex scheme_current sub_processor CPMAXCORES 50
Powercfg -setactive scheme_current
```

Processor performance core parking utility distribution

Utility Distribution is an algorithmic optimization in Windows Server 2016 that is designed to improve power efficiency for some workloads. It tracks unmovable CPU activity (that is, DPCs, interrupts, or strictly affinitized threads), and it predicts the future work on each processor based on the assumption that any movable work can

be distributed equally across all unparked cores.

Utility Distribution is enabled by default for the Balanced power plan for some processors. It can reduce processor power consumption by lowering the requested CPU frequencies of workloads that are in a reasonably steady state. However, Utility Distribution is not necessarily a good algorithmic choice for workloads that are subject to high activity bursts or for programs where the workload quickly and randomly shifts across processors.

For such workloads, we recommend disabling Utility Distribution by using the following commands:

```
Powercfg -setacvalueindex scheme_current sub_processor DISTRIBUTEUTIL 0
Powercfg -setactive scheme_current
```

Additional References

- Server Hardware Performance Considerations
- Server Hardware Power Considerations
- Processor Power Management (PPM) tuning for the Windows Server balanced power plan

Processor Power Management (PPM) Tuning for the Windows Server Balanced Power Plan

12/16/2022 • 12 minutes to read • Edit Online

Starting with Windows Server 2008, Windows Server provides three power plans: **Balanced**, **High Performance**, and **Power Saver**. The **Balanced** power plan is the default choice that aims to give the best energy efficiency for a set of typical server workloads. This topic describes the workloads that have been used to determine the default settings for the **Balanced** scheme for the past several releases of Windows.

If you run a server system that has dramatically different workload characteristics or performance and power requirements than these workloads, you might want to consider tuning the default power settings (that is, create a custom power plan). One source of useful tuning information is the Server Hardware Power Considerations. Alternately, you may decide that the **High Performance** power plan is the right choice for your environment, recognizing that you will likely take a significant energy hit in exchange for some level of increased responsiveness.

> **IMPORTANT**
> You should leverage the power policies that are included with Windows Server unless you have a specific need to create a custom one and have a very good understanding that your results will vary depending on the characteristics of your workload.

Windows Processor Power Tuning Methodology

Tested workloads

Workloads are selected to cover a best-effort set of "typical" Windows Server workloads. Obviously this set is not intended to be representative of the entire breadth of real-world server environments.

The tuning in each power policy is data driven by the following five workloads started from Windows Server 2008

- **IIS Web Server workload**

  A Microsoft internal benchmark called Web Fundamentals is used to optimize the energy efficiency of platforms running IIS Web Server. The setup contains a web server and multiple clients that simulate the web access traffic. The distribution of dynamic, static hot (in-memory), and static cold (disk access required) web pages is based on statistical studies of production servers. To push the server's CPU cores to full utilization (one end of the tested spectrum), the setup needs sufficiently fast network and disk resources.

- **SQL Server Database workload**

  The TPC-E benchmark is a popular benchmark for database performance analysis. It is used to generate an OLTP workload for PPM tuning optimizations. This workload has significant disk I/O, and hence has a high performance requirement for the storage system and memory size.

- **File Server workload**

  A Microsoft-developed benchmark called FSCT is used to generate an SMB file server workload. It creates a large file set on the server and uses many client systems (actual or virtualized) to generate file open, close, read, and write operations. The operation mix is based on statistical studies of production servers. It

stresses CPU, disk, and network resources.

- **SPECpower – JAVA workload**

  SPECpower_ssj2008 is the first industry-standard SPEC benchmark that jointly evaluates power and performance characteristics. It is a server-side Java workload with varying CPU load levels. It doesn't require many disk or network resources, but it has certain requirements for memory bandwidth. Almost all of the CPU activity is performed in user-mode; kernel-mode activity does not have much impact on the benchmarks' power and performance characteristics except for the power management decisions.

- **Application Server workload**

  The SAP-SD benchmark is used to generate an application server workload. A two-tier setup is used, with the database and the application server on the same server host. This workload also utilizes response time as a performance metric, which differs from other tested workloads. Thus it is used to verify the impact of PPM parameters on responsiveness. Nevertheless, it is not intended to be representative of all latency-sensitive production workloads.

All of the benchmarks except SPECpower were originally designed for performance analysis and were therefore created to run at peak load levels. However, medium to light load levels are more common for real-world production servers and are more interesting for **Balanced** plan optimizations. We intentionally run the benchmarks at varying load levels from 100% down to 10% (in 10% steps) by using various throttling methods (for example, by reducing the number of active users/clients).

The above workloads use throughput as the performance metric for tuning. During the steady state, throughput does not change with varying utilizations until the system is overloaded (~100% utilization). As a result, the Balanced power plan favors power quite a lot with minimizing processor frequency and maximizing utilization. Starting from Windows Server 2016, the requirement of quick response time has dramatically increased. Even though Microsoft suggested the users switch to the High Performance power plan when they need quick response time, some users do not want to lose the power benefit during light to medium load levels. Hence, Windows Server PPM tuning also includes response time sensitive workloads for tuning.

- **GeekBench 3**

  GeekBench 3 is a cross-platform processor benchmark that separates the scores for single-core and multi-core performance. It simulates a set of workloads including integer workloads (encryptions, compressions, image processing, etc.), floating point workloads (modeling, fractal, image sharpening, image blurring, etc.) and memory workloads (streaming).

  Response time is a major measure in its score calculation. In our tested system, Windows Server 2008 default Balanced power plan has ~18% regression in single-core tests and ~40% regression in multi-core tests compared to the **High Performance** power plan. Windows Server 2016 removes these regressions.

- **DiskSpd**

  Diskspd is a command-line tool for storage benchmarking developed by Microsoft. It is widely used to generate various requests against storage systems for the storage performance analysis.

  We set up a [Failover Cluster], and used Diskspd to generate random and sequential, and read and write IOs to the local and remote storage systems with different IO sizes. Our tests show that the IO response time is sensitive to processor frequency under different power plans. The default Windows Server 2008 Balanced power plan could double the response time of that from the **High Performance** power plan under certain workloads. Windows Server 2016 Balance power plan removes most of the regressions.

> **IMPORTANT**
>
> Starting from Intel [Broadwell] processors running Windows Server 2016, most of the processor power management decisions are made in the processor instead of OS level to achieve quicker adaption to the workload changes. The legacy PPM parameters used by OS have minimal impact on the actual frequency decisions, except telling the processor if it should favor power or performance, or capping the minimal and maximum frequencies. Hence, the PPM tuning results mentioned here are only observed on the pre-Broadwell systems.

Hardware configurations

For each release of Windows, the most current production servers are used in the power plan analysis and optimization process. In some cases, the tests were performed on pre-production systems whose release schedule matched that of the next Windows release.

Given that most servers are sold with 1 to 4 processor sockets, and since scale-up servers are less likely to have energy efficiency as a primary concern, the power plan optimization tests are primarily run on 2-socket and 4-socket systems. The amount of RAM, disk, and network resources for each test are chosen to allow each system to run all the way up to its full capacity, while taking into account the cost restrictions that would normally be in place for real-world server environments, such as keeping the configurations reasonable.

> **IMPORTANT**
>
> Even though the system can run at its peak load, we typically optimize for lower load levels, since servers that consistently run at their peak load levels would be well-advised to use the **High Performance** power plan unless energy efficiency is a high priority.

Metrics

All of the tested benchmarks use throughput as the performance metric. Response Time is considered as an SLA requirement for these workloads (except for SAP, where it is a primary metric). For example, a benchmark run is considered "valid" if the mean or maximum response time is less than certain value.

Therefore, the PPM tuning analysis also uses throughput as its performance metric. At the highest load level (100% CPU utilization), our goal is that the throughput should not decrease more than a few percent due to power management optimizations. But the primary consideration is to maximize the power efficiency (as defined below) at medium and low load levels.

$$Power\ Efficiency = \frac{Throughput}{System\ Power\ in\ Watts}$$

Running the CPU cores at lower frequencies reduces energy consumption. However, lower frequencies typically decrease throughput and increase response time. For the **Balanced** power plan, there is an intentional tradeoff of responsiveness and power efficiency. The SAP workload tests, as well as the response time SLAs on the other workloads, make sure that the response time increase doesn't exceed certain threshold (5% as an example) for these specific workloads.

> **NOTE**
>
> If the workload is very sensitive to response time, the system should either switch to the **High Performance** power plan or change **Balanced** power plan to very aggressively increase frequency when it is running.

Windows Server Balanced Power Plan Default Parameters

Starting from Intel Broadwell processors running Windows Server 2016, Windows Server power management

uses Intel's Hardware Controlled P-states (HWP) as default on Intel systems. HWP is a new capability for a cooperative hardware and software performance control. When HWP is enabled, CPU monitors activity and scalability, and selects frequency at hardware time scale. OS is no longer required to monitor activity and select frequency at regular intervals. Switching to HWP has several benefits such as quick response, better knowledge of the hardware power efficiency of processors and other components under TDP.

For HWP system, Windows still has the option to set the minimum and maximum processor states to provide constraints. It also can use **Energy performance preference (EPP)** parameter to set balance between power and performance. Lower value favors performance, and higher value favors power. The default value 50 that is to balance power and performance.

PARAMETER	WINDOWS SERVER 2012R2 AND BEFORE	WINDOWS SERVER 2016 AND AFTER
HWP Enabled	N/A	Intel Broadwell+
Energy performance preference	N/A	50

For Intel pre-Broadwell systems or any systems that don't have HWP support (for example, AMD servers), Windows is still in full control and determines processor frequency based on the PPM parameters. The default PPM parameters in Windows Server 2012R2 favor power too much that could significantly impact the workload performance, especially to bursty workload. Four PPM parameters were changed in Windows Server 2016 RS2 to let the frequency increase faster around medium load level.

PARAMETER	WINDOWS SERVER 2016 (RS1) AND BEFORE	WINDOWS SERVER 2016 (RS2) AND AFTER
Processor performance increase threshold	90	60
Processor performance decrease threshold	80	40
Processor performance increase time	3	1
Processor performance increase policy	Single	Ideal

The CPU utilization-based power management algorithms might hurt the latency of IO or network-intensive workloads. A logical processor could be idle while waiting for IO completion or network packets, which makes the overall CPU utilization low. To resolve this issue, Windows Server 2019 automatically detects the IO responsiveness period and raises the frequency floor to a higher level. The behavior can be tuned by the following parameters no matter if the system uses HWP or not.

PARAMETER	BEFORE WINDOWS SERVER 2019	WINDOWS SERVER 2019 AND AFTER
Processor responsiveness override enable threshold	N/A	10
Processor responsiveness override disable threshold	N/A	5
Processor responsiveness override enable time	N/A	1

PARAMETER	BEFORE WINDOWS SERVER 2019	WINDOWS SERVER 2019 AND AFTER
Processor responsiveness override disable time	N/A	3
Processor responsiveness override energy performance preference ceiling	N/A	100
Processor responsiveness override performance floor	N/A	100

Customized Tuning Suggestions

If your primary workload characteristics differ significantly from the five workloads used for the default **Balanced** power plan PPM tuning, you can experiment by altering one or more PPM parameters to find the best fit for your environment.

Due to the number and complexity of parameters, this may be a challenging task, but if you are looking for the best tradeoff between energy consumption and workload efficacy for your particular environment, it may be worth the effort.

The complete set of tunable PPM parameters can be found in Processor power management tuning. Some of the simplest power parameters to start with could be:

For HWP enabled system:

- **Energy performance preference** – larger values favors power more than performance

For Non-HWP system:

- **Processor Performance Increase Threshold and Processor Performance Increase Time** – larger values slow the perf response to increased activity
- **Processor Performance Decrease Threshold** – large values quicken the power response to idle periods
- **Processor Performance Decrease Time** – larger values more gradually decrease perf during idle periods
- **Processor Performance Increase Policy** – the "Single" policy slows the perf response to increased and sustained activity; the "Rocket" policy reacts quickly to increased activity
- **Processor Performance Decrease Policy** – the "Single" policy more gradually decreases perf over longer idle periods; the "Rocket" policy drops power very quickly when entering an idle period

> **IMPORTANT**
> Before starting any experiments, you should first understand your workloads, which will help you make the right PPM parameter choices and reduce the tuning effort.

Understand high-level performance and power requirements

If your workload is "real time" (for example, susceptible to glitching or other visible end-user impacts) or has very tight responsiveness requirement (for example, a stock brokerage), and if energy consumption is not a primary criteria for your environment, you should probably just switch to the **High Performance** power plan. Otherwise, you should understand the response time requirements of your workloads, and then tune the PPM parameters for the best possible power efficiency that still meets those requirements.

Understand underlying workload characteristics

You should know your workloads and design the experiment parameter sets for tuning. For example, if frequencies of the CPU cores need to be ramped fast (perhaps you have a bursty workload with significant idle periods, but you need very quick responsiveness when a new transaction comes along), then the processor performance increase policy might need to be set to "rocket" (which, as the name implies, shoots the CPU core frequency to its maximum value rather than stepping it up over a period of time).

If your workload is very bursty, the PPM check interval can be reduced to make the CPU frequency start stepping up sooner after a burst arrives. If your workload doesn't have high thread concurrency, then core parking can be enabled to force the workload to execute on a smaller number of cores, which could also potentially improve processor cache hit ratios.

If you just want to increase CPU frequencies at medium utilization levels (i.e., not light workload levels), then processor performance increase/decrease thresholds can be adjusted to not react until certain levels of activity are observed.

Understand periodic behaviors

There may be different performance requirements for daytime and nighttime or over the weekends, or there may be different workloads that run at different times. In this case, one set of PPM parameters might not be optimal for all time periods. Since multiple custom power plans can be devised, it is possible to even tune for different time periods and switch between power plans through scripts or other means of dynamic system configuration.

Again, this adds to the complexity of the optimization process, so it is a question of how much value will be gained from this type of tuning, which will likely need to be repeated when there are significant hardware upgrades or workload changes.

This is why Windows provides a **Balanced** power plan in the first place, because in many cases it is probably not worth the effort of hand-tuning for a specific workload on a specific server.

See Also

- Server Hardware Performance Considerations
- Server Hardware Power Considerations
- Overview about power and performance tuning for the Windows Server

Performance tuning Active Directory Servers

12/16/2022 • 2 minutes to read • Edit Online

Performance tuning Active Directory is focused on two goals:

- Active Directory is optimally configured to service the load in the most efficient manner possible
- Workloads submitted to Active Directory should be as efficient as possible

This requires proper attention to three separate areas:

- Proper capacity planning – ensuring sufficient hardware is in place to support existing load
- Server side tuning – configuring domain controllers to handle the load as efficiently as possible
- Active Directory client/application tuning – ensuring that clients and applications are using Active Directory in an optimal fashion

Start with capacity planning

Properly deploying a sufficient number of domain controllers, in the right domain, in the right locales, and to accommodate redundancy is critical to ensuring servicing client requests in a timely fashion. This is an in-depth topic and outside of the scope of this guide. Readers are encouraged start to their Active Directory performance tuning by reading and understanding the recommendations and guidance found in Capacity planning for Active Directory Domain Services.

> **IMPORTANT**
>
> Proper configuration and sizing of Active Directory has a significant potential impact on overall system and workload performance. Readers are highly encouraged to start by reading Capacity planning for Active Directory Domain Services.

Updates and evolving recommendations

Massive improvements in both Active Directory and client performance optimizations have occurred over the last several generations of the operating system and these efforts continue. We recommend that the most current versions of the platform be deployed to get the benefits, including:

- Increased reliability
- Better performance
- Better logging and tools to troubleshoot

However, we realize that this takes time and many environments are running in a scenario where 100% adoption of the most current platform is impossible. Some improvements have been added to older versions of the platform and we'll continue to add more.

We encourage you to stay up to date on the latest news, guidance and best practices for managing ADDS by following our team blog, "Ask the Directory Services Team".

Additional References

- Capacity planning for AD DS
- Hardware considerations
- Memory usage considerations

- LDAP considerations
- Proper placement of domain controllers and site considerations
- Troubleshooting AD DS performance

Capacity planning for Active Directory Domain Services

12/16/2022 • 92 minutes to read • Edit Online

This topic is originally written by Ken Brumfield, Program Manager at Microsoft, and provides recommendations for capacity planning for Active Directory Domain Services (AD DS).

Goals of capacity planning

Capacity planning is not the same as troubleshooting performance incidents. They are closely related, but quite different. The goals of capacity planning are:

- Properly implement and operate an environment
- Minimize the time spent troubleshooting performance issues.

In capacity planning, an organization might have a baseline target of 40% processor utilization during peak periods in order to meet client performance requirements and accommodate the time necessary to upgrade the hardware in the datacenter. Whereas, to be notified of abnormal performance incidents, a monitoring alert threshold might be set at 90% over a 5 minute interval.

The difference is that when a capacity management threshold is continually exceeded (a one-time event is not a concern), adding capacity (that is, adding in more or faster processors) would be a solution or scaling the service across multiple servers would be a solution. Performance alert thresholds indicate that client experience is currently suffering and immediate steps are needed to address the issue.

As an analogy: capacity management is about preventing a car accident (defensive driving, making sure the brakes are working properly, and so on) whereas performance troubleshooting is what the police, fire department, and emergency medical professionals do after an accident. This is about "defensive driving," Active Directory-style.

Over the last several years, capacity planning guidance for scale-up systems has changed dramatically. The following changes in system architectures have challenged fundamental assumptions about designing and scaling a service:

- 64-bit server platforms
- Virtualization
- Increased attention to power consumption
- SSD storage
- Cloud scenarios

Additionally, the approach is shifting from a server-based capacity planning exercise to a service-based capacity planning exercise. Active Directory Domain Services (AD DS), a mature distributed service that many Microsoft and third-party products use as a backend, becomes one the most critical products to plan correctly to ensure the necessary capacity for other applications to run.

Baseline requirements for capacity planning guidance

Throughout this article, the following baseline requirements are expected:

- Readers have read and are familiar with Performance Tuning Guidelines for Windows Server 2012 R2.
- The Windows Server platform is an x64 based architecture. But even if your Active Directory environment is installed on Windows Server 2003 x86 (now beyond the end of the support lifecycle) and has a directory

information tree (DIT) that is less 1.5 GB in size and that can easily be held in memory, the guidelines from this article are still applicable.

- Capacity planning is a continuous process and you should regularly review how well the environment is meeting expectations.
- Optimization will occur over multiple hardware lifecycles as hardware costs change. For example, memory becomes cheaper, the cost per core decreases, or the price of different storage options change.
- Plan for the peak busy period of the day. It is recommended to look at this in either 30 minute or hour intervals. Anything greater may hide the actual peaks and anything less may be distorted by "transient spikes."
- Plan for growth over the course of the hardware lifecycle for the enterprise. This may include a strategy of upgrading or adding hardware in a staggered fashion, or a complete refresh every three to five years. Each will require a "guess" as how much the load on Active Directory will grow. Historical data, if collected, will help with this assessment.
- Plan for fault tolerance. Once an estimate N is derived, plan for scenarios that include $N-1$, $N-2$, $N-x$.
    - Add in additional servers according to organizational need to ensure that the loss of a single or multiple servers does not exceed maximum peak capacity estimates.
    - Also consider that the growth plan and fault tolerance plan need to be integrated. For example, if one DC is required to support the load, but the estimate is that the load will be doubled in the next year and require two DCs total, there will not be enough capacity to support fault tolerance. The solution would be to start with three DCs. One could also plan to add the third DC after 3 or 6 months if budgets are tight.

      > **NOTE**
      > Adding Active Directory-aware applications might have a noticeable impact on the DC load, whether the load is coming from the application servers or clients.

Three-step process for the capacity planning cycle

In capacity planning, first decide what quality of service is needed. For example, a core datacenter supports a higher level of concurrency and requires more consistent experience for users and consuming applications, which requires greater attention to redundancy and minimizing system and infrastructure bottlenecks. In contrast, a satellite location with a handful of users does not need the same level of concurrency or fault tolerance. Thus, the satellite office might not need as much attention to optimizing the underlying hardware and infrastructure, which may lead to cost savings. All recommendations and guidance herein are for optimal performance, and can be selectively relaxed for scenarios with less demanding requirements.

The next question is: virtualized or physical? From a capacity planning perspective, there is no right or wrong answer; there is only a different set of variables to work with. Virtualization scenarios come down to one of two options:

- "Direct mapping" with one guest per host (where virtualization exists solely to abstract the physical hardware from the server)
- "Shared host"

Testing and production scenarios indicate that the "direct mapping" scenario can be treated identically to a physical host. "Shared host," however, introduces a number of considerations spelled out in more detail later. The "shared host" scenario means that AD DS is also competing for resources, and there are penalties and tuning considerations for doing so.

With these considerations in mind, the capacity planning cycle is an iterative three-step process:

1. Measure the existing environment, determine where the system bottlenecks currently are, and get

environmental basics necessary to plan the amount of capacity needed.
2. Determine the hardware needed according to the criteria outlined in step 1.
3. Monitor and validate that the infrastructure as implemented is operating within specifications. Some data collected in this step becomes the baseline for the next cycle of capacity planning.

Applying the process

To optimize performance, ensure these major components are correctly selected and tuned to the application loads:

1. Memory
2. Network
3. Storage
4. Processor
5. Net Logon

The basic storage requirements of AD DS and the general behavior of well written client software allow environments with as many as 10,000 to 20,000 users to forego heavy investment in capacity planning with regards to physical hardware, as almost any modern server class system will handle the load. That said, the following table summarizes how to evaluate an existing environment in order to select the right hardware. Each component is analyzed in detail in subsequent sections to help AD DS administrators evaluate their infrastructure using baseline recommendations and environment-specific principals.

In general:

- Any sizing based on current data will only be accurate for the current environment.
- For any estimates, expect demand to grow over the lifecycle of the hardware.
- Determine whether to oversize today and grow into the larger environment, or add capacity over the lifecycle.
- For virtualization, all the same capacity planning principals and methodologies apply, except that the overhead of the virtualization needs to be added to anything domain related.
- Capacity planning, like anything that attempts to predict, is NOT an accurate science. Do not expect things to calculate perfectly and with 100% accuracy. The guidance here is the leanest recommendation; add in capacity for additional safety and continuously validate that the environment remains on target.

Data collection summary tables

New environment

COMPONENT	ESTIMATES
Storage/Database Size	40 KB to 60 KB for each user
RAM	Database Size Base operating system recommendations Third-party applications
Network	1 GB
CPU	1000 concurrent users for each core

High-level evaluation criteria

COMPONENT	EVALUATION CRITERIA	PLANNING CONSIDERATIONS

COMPONENT	EVALUATION CRITERIA	PLANNING CONSIDERATIONS
Storage/Database size	The section entitled "To activate logging of disk space that is freed by defragmentation" in Storage Limits	

Storage/ Database performance	• "LogicalDisk(*<NTDS Database Drive>*)\Avg Disk sec/Read," "LogicalDisk(*<NTDS Database Drive>*)\Avg Disk sec/Write," "LogicalDisk(*<NTDS Database Drive>*)\Avg Disk sec/Transfer"
	• "LogicalDisk(*<NTDS Database Drive>*)\Reads/sec," "LogicalDisk(*<NTDS Database Drive>*)\Writes/sec," "LogicalDisk(*<NTDS Database Drive>*)\Transfers/sec"
		• Storage has two concerns to address
		  ◦ Space available, which with the size of today's spindle based and SSD based storage is irrelevant for most AD environments.
		  ◦ Input/Output (IO) Operations available – In many environments, this is often overlooked. But it is important to evaluate only environments where there is not enough RAM to load the entire NTDS Database into memory.
		• Storage can be a complex topic and should involve hardware vendor expertise for proper sizing. Particularly with more complex scenarios such as SAN, NAS, and iSCSI scenarios. However, in general, cost per Gigabyte of storage is often in direct opposition to cost per IO:
		  ◦ RAID 5 has lower cost per Gigabyte than Raid 1, but Raid 1 has lower cost per IO
		  ◦ Spindle-based hard drives have lower cost per Gigabyte, but SSDs have a lower cost per IO
		• After a restart of the computer or the Active Directory Domain Services service, the Extensible Storage Engine (ESE) cache is empty and performance will be disk bound while the cache warms.
		• In most environments AD is read intensive I/O in a random pattern to disks, negating much of the benefit of caching and read optimization strategies. Plus, AD has a way larger cache in memory than most storage system caches.

COMPONENT	EVALUATION CRITERIA	PLANNING CONSIDERATIONS
RAM	• Database size • Base operating system recommendations • Third-party applications	• Storage is the slowest component in a computer. The more that can be resident in RAM, the less it is necessary to go to disk. • Ensure enough RAM is allocated to store the operating system, Agents (antivirus, backup, monitoring), NTDS Database and growth over time. • For environments where maximizing the amount of RAM is not cost effective (such as a satellite locations) or not feasible (DIT is too large), reference the Storage section to ensure that storage is properly sized.
Network	• "Network Interface(*)\Bytes Received/sec" • "Network Interface(*)\Bytes Sent/sec"	• In general, traffic sent from a DC far exceeds traffic sent to a DC. • As a switched Ethernet connection is full-duplex, inbound and outbound network traffic need to be sized independently. • Consolidating the number of DCs will increase the amount of bandwidth used to send responses back to client requests for each DC, but will be close enough to linear for the site as a whole. • If removing satellite location DCs, don't forget to add the bandwidth for the satellite DC into the hub DCs as well as use that to evaluate how much WAN traffic there will be.

COMPONENT	EVALUATION CRITERIA	PLANNING CONSIDERATIONS
CPU	• "Logical Disk(*<NTDS Database Drive>*)\Avg Disk sec/Read" • "Process(lsass)\% Processor Time"	• After eliminating storage as a bottleneck, address the amount of compute power needed. • While not perfectly linear, the number of processor cores consumed across all servers within a specific scope (such as a site) can be used to gauge how many processors are necessary to support the total client load. Add the minimum necessary to maintain the current level of service across all the systems within the scope. • Changes in processor speed, including power management related changes, impact numbers derived from the current environment. Generally, it is impossible to precisely evaluate how going from a 2.5 GHz processor to a 3 GHz processor will reduce the number of CPUs needed.
NetLogon	• "Netlogon(*)\Semaphore Acquires"* • "Netlogon()\Semaphore Timeouts" • "Netlogon(*)\Average Semaphore Hold Time"	• Net Logon Secure Channel/MaxConcurrentAPI only affects environments with NTLM authentications and/or PAC Validation. PAC Validation is on by default in operating system versions before Windows Server 2008. This is a client setting, so the DCs will be impacted until this is turned off on all client systems. • Environments with significant cross trust authentication, which includes intra-forest trusts, have greater risk if not sized properly. • Server consolidations will increase concurrency of cross-trust authentication. • Surges need to be accommodated, such as cluster fail-overs, as users re-authenticate en masse to the new cluster node. • Individual client systems (such as a cluster) might need tuning too.

Planning

For a long time, the community's recommendation for sizing AD DS has been to "put in as much RAM as the database size." For the most part, that recommendation is all that most environments needed to be concerned about. But the ecosystem consuming AD DS has gotten much bigger, as have the AD DS environments themselves, since its introduction in 1999. Although the increase in compute power and the switch from x86 architectures to x64 architectures has made the subtler aspects of sizing for performance irrelevant to a larger set of customers running AD DS on physical hardware, the growth of virtualization has reintroduced the tuning concerns to a larger audience than before.

The following guidance is thus about how to determine and plan for the demands of Active Directory as a service regardless of whether it is deployed in a physical, a virtual/physical mix, or a purely virtualized scenario. As such, we will break down the evaluation to each of the four main components: storage, memory, network, and processor. In short, in order to maximize performance on AD DS, the goal is to get as close to processor bound as possible.

RAM

Simply, the more that can be cached in RAM, the less it is necessary to go to disk. To maximize the scalability of the server the minimum amount of RAM should be the sum of the current database size, the total SYSVOL size, the operating system recommended amount, and the vendor recommendations for the agents (antivirus, monitoring, backup, and so on). An additional amount should be added to accommodate growth over the lifetime of the server. This will be environmentally subjective based on estimates of database growth based on environmental changes.

For environments where maximizing the amount of RAM is not cost effective (such as a satellite locations) or not feasible (DIT is too large), reference the Storage section to ensure that storage is properly designed.

A corollary that comes up in the general context in sizing memory is sizing of the page file. In the same context as everything else memory related, the goal is to minimize going to the much slower disk. Thus the question should go from, "how should the page file be sized?" to "how much RAM is needed to minimize paging?" The answer to the latter question is outlined in the rest of this section. This leaves most of the discussion for sizing the page file to the realm of general operating system recommendations and the need to configure the system for memory dumps, which are unrelated to AD DS performance.

Evaluating

The amount of RAM that a domain controller (DC) needs is actually a complex exercise for these reasons:

- High potential for error when trying to use an existing system to gauge how much RAM is needed as LSASS will trim under memory pressure conditions, artificially deflating the need.
- The subjective fact that an individual DC only needs to cache what is "interesting" to its clients. This means that the data that needs to be cached on a DC in a site with only an Exchange server will be very different than the data that needs to be cached on a DC that only authenticates users.
- The labor to evaluate RAM for each DC on a case-by-case basis is prohibitive and changes as the environment changes.
- The criteria behind the recommendation will help to make informed decisions:
- The more that can be cached in RAM, the less it is necessary to go to disk.
- Storage is by far the slowest component of a computer. Access to data on spindle-based and SSD storage media is on the order of 1,000,000x slower than access to data in RAM.

Thus, in order to maximize the scalability of the server, the minimum amount of RAM is the sum of the current database size, the total SYSVOL size, the operating system recommended amount, and the vendor recommendations for the agents (antivirus, monitoring, backup, and so on). Add additional amounts to accommodate growth over the lifetime of the server. This will be environmentally subjective based on estimates of database growth. However, for satellite locations with a small set of end users, these requirements can be relaxed as these sites will not need to cache as much to service most of the requests.

For environments where maximizing the amount of RAM is not cost effective (such as a satellite locations) or not feasible (DIT is too large), reference the Storage section to ensure that storage is properly sized.

> **NOTE**
>
> A corollary while sizing memory is sizing of the page file. Because the goal is to minimize going to the much slower disk, the question goes from "how should the page file be sized?" to "how much RAM is needed to minimize paging?" The answer to the latter question is outlined in the rest of this section. This leaves most of the discussion for sizing the page file to the realm of general operating system recommendations and the need to configure the system for memory dumps, which are unrelated to AD DS performance.

Virtualization considerations for RAM

Avoid memory over-commit at the host. The fundamental goal behind optimizing the amount of RAM is to minimize the amount of time spent going to disk. In virtualization scenarios, the concept of memory over-commit exists where more RAM is allocated to the guests then exists on the physical machine. This in and of itself is not a problem. It becomes a problem when the total memory actively used by all the guests exceeds the amount of RAM on the host and the underlying host starts paging. Performance becomes disk-bound in cases where the domain controller is going to the NTDS.dit to get data, or the domain controller is going to the page file to get data, or the host is going to disk to get data that the guest thinks is in RAM.

Calculation summary example

COMPONENT	ESTIMATED MEMORY (EXAMPLE)
Base operating system recommended RAM (Windows Server 2008)	2 GB
LSASS internal tasks	200 MB
Monitoring agent	100 MB
Antivirus	100 MB
Database (Global Catalog)	8.5 GB
Cushion for backup to run, administrators to log on without impact	1 GB
Total	12 GB

Recommended: 16 GB

Over time, the assumption can be made that more data will be added to the database and the server will probably be in production for 3 to 5 years. Based on an estimate of growth of 33%, 16 GB would be a reasonable amount of RAM to put in a physical server. In a virtual machine, given the ease with which settings can be modified and RAM can be added to the VM, starting at 12 GB with the plan to monitor and upgrade in the future is reasonable.

Network

Evaluating

This section is less about evaluating the demands regarding replication traffic, which is focused on traffic traversing the WAN and is thoroughly covered in Active Directory Replication Traffic, than it is about evaluating total bandwidth and network capacity needed, inclusive of client queries, Group Policy applications, and so on.

For existing environments, this can be collected by using performance counters "Network Interface(*)\Bytes Received/sec," and "Network Interface(*)\Bytes Sent/sec." Sample intervals for Network Interface counters in either 15, 30, or 60 minutes. Anything less will generally be too volatile for good measurements; anything greater will smooth out daily peeks excessively.

> **NOTE**
>
> Generally, the majority of network traffic on a DC is outbound as the DC responds to client queries. This is the reason for the focus on outbound traffic, though it is recommended to evaluate each environment for inbound traffic also. The same approaches can be used to address and review inbound network traffic requirements. For more information, see Knowledge Base article 929851: The default dynamic port range for TCP/IP has changed in Windows Vista and in Windows Server 2008.

Bandwidth needs

Planning for network scalability covers two distinct categories: the amount of traffic and the CPU load from the network traffic. Each of these scenarios is straight-forward compared to some of the other topics in this article.

In evaluating how much traffic must be supported, there are two unique categories of capacity planning for AD DS in terms of network traffic. The first is replication traffic that traverses between domain controllers and is covered thoroughly in the reference Active Directory Replication Traffic and is still relevant to current versions of AD DS. The second is the intrasite client-to-server traffic. One of the simpler scenarios to plan for, intrasite traffic predominantly receives small requests from clients relative to the large amounts of data sent back to the clients. 100 MB will generally be adequate in environments up to 5,000 users per server, in a site. Using a 1 GB network adapter and Receive Side Scaling (RSS) support is recommended for anything above 5,000 users. To validate this scenario, particularly in the case of server consolidation scenarios, look at Network Interface(*)\Bytes/sec across all the DCs in a site, add them together, and divide by the target number of domain controllers to ensure that there is adequate capacity. The easiest way to do this is to use the "Stacked Area" view in Windows Reliability and Performance Monitor (formerly known as Perfmon), making sure all of the counters are scaled the same.

Consider the following example (also known as, a really, really complex way to validate that the general rule is applicable to a specific environment). The following assumptions are made:

- The goal is to reduce the footprint to as few servers as possible. Ideally, one server will carry the load and an additional server is deployed for redundancy (*N* + 1 scenario).
- In this scenario, the current network adapter supports only 100 MB and is in a switched environment. The maximum target network bandwidth utilization is 60% in an N scenario (loss of a DC).
- Each server has about 10,000 clients connected to it.

Knowledge gained from the data in the chart (Network Interface(*)\Bytes Sent/sec):

1. The business day starts ramping up around 5:30 and winds down at 7:00 PM.
2. The peak busiest period is from 8:00 AM to 8:15 AM, with greater than 25 Bytes sent/sec on the busiest DC.

   > **NOTE**
   >
   > All performance data is historical. So the peak data point at 8:15 indicates the load from 8:00 to 8:15.

3. There are spikes before 4:00 AM, with more than 20 Bytes sent/sec on the busiest DC, which could indicate either load from different time zones or background infrastructure activity, such as backups. Since the peak at 8:00 AM exceeds this activity, it is not relevant.
4. There are five Domain Controllers in the site.
5. The max load is about 5.5 MB/s per DC, which represents 44% of the 100 MB connection. Using this data, it can be estimated that the total bandwidth needed between 8:00 AM and 8:15 AM is 28 MB/s.

> **NOTE**
>
> Be careful with the fact that Network Interface sent/receive counters are in bytes and network bandwidth is measured in bits. 100 MB ÷ 8 = 12.5 MB, 1 GB ÷ 8 = 128 MB.

Conclusions:

1. This current environment does meet the N+1 level of fault tolerance at 60% target utilization. Taking one system offline will shift the bandwidth per server from about 5.5 MB/s (44%) to about 7 MB/s (56%).
2. Based on the previously stated goal of consolidating to one server, this both exceeds the maximum target utilization and theoretically the possible utilization of a 100 MB connection.
3. With a 1 GB connection this will represent 22% of the total capacity.
4. Under normal operating conditions in the *N* + 1 scenario, client load will be relatively evenly distributed at about 14 MB/s per server or 11% of total capacity.
5. To ensure that capacity is adequate during unavailability of a DC, the normal operating targets per server would be about 30% network utilization or 38 MB/s per server. Failover targets would be 60% network utilization or 72 MB/s per server.

In short, the final deployment of systems must have a 1 GB network adapter and be connected to a network infrastructure that will support said load. A further note is that given the amount of network traffic generated, the CPU load from network communications can have a significant impact and limit the maximum scalability of AD DS. This same process can be used to estimate the amount of inbound communication to the DC. But given the predominance of outbound traffic relative to inbound traffic, it is an academic exercise for most environments. Ensuring hardware support for RSS is important in environments with greater than 5,000 users per server. For scenarios with high network traffic, balancing of interrupt load can be a bottleneck. This can be detected by Processor(*)% Interrupt Time being unevenly distributed across CPUs. RSS enabled NICs can mitigate this limitation and increase scalability.

> **NOTE**
>
> A similar approach can be used to estimate the additional capacity necessary when consolidating data centers, or retiring a domain controller in a satellite location. Simply collect the outbound and inbound traffic to clients and that will be the amount of traffic that will now be present on the WAN links.
>
> In some cases, you might experience more traffic than expected because traffic is slower, such as when certificate checking fails to meet aggressive time-outs on the WAN. For this reason, WAN sizing and utilization should be an iterative, ongoing process.

Virtualization considerations for network bandwidth

It is easy to make recommendations for a physical server: 1 GB for servers supporting greater than 5000 users. Once multiple guests start sharing an underlying virtual switch infrastructure, additional attention is necessary to ensure that the host has adequate network bandwidth to support all the guests on the system, and thus requires the additional rigor. This is nothing more than an extension of ensuring the network infrastructure into the host machine. This is regardless whether the network is inclusive of the domain controller running as a virtual machine guest on a host with the network traffic going over a virtual switch, or whether connected directly to a physical switch. The virtual switch is just one more component where the uplink needs to support the amount of data being transmitted. Thus the physical host physical network adapter linked to the switch should be able to support the DC load plus all other guests sharing the virtual switch connected to the physical network adapter.

Calculation summary example

SYSTEM	PEAK BANDWIDTH
DC 1	6.5 MB/s
DC 2	6.25 MB/s
DC 3	6.25 MB/s
DC 4	5.75 MB/s
DC 5	4.75 MB/s
Total	28.5 MB/s

Recommended: 72 MB/s (28.5 MB/s divided by 40%)

TARGET SYSTEM(S) COUNT	TOTAL BANDWIDTH (FROM ABOVE)
2	28.5 MB/s
Resulting normal behavior	28.5 ÷ 2 = 14.25 MB/s

As always, over time the assumption can be made that client load will increase and this growth should be planned for as best as possible. The recommended amount to plan for would allow for an estimated growth in network traffic of 50%.

Storage

Planning storage constitutes two components:

- Capacity, or storage size
- Performance

A great amount of time and documentation is spent on planning capacity, leaving performance often completely overlooked. With current hardware costs, most environments are not large enough that either of these is actually a concern, and the recommendation to "put in as much RAM as the database size" usually covers the rest, though it may be overkill for satellite locations in larger environments.

Sizing

Evaluating for storage

Compared to 13 years ago when Active Directory was introduced, a time when 4 GB and 9 GB drives were the most common drive sizes, sizing for Active Directory is not even a consideration for all but the largest environments. With the smallest available hard drive sizes in the 180 GB range, the entire operating system, SYSVOL, and NTDS.dit can easily fit on one drive. As such, it is recommended to deprecate heavy investment in this area.

The only recommendation for consideration is to ensure that 110% of the NTDS.dit size is available in order to enable defrag. Additionally, accommodations for growth over the life of the hardware should be made.

The first and most important consideration is evaluating how large the NTDS.dit and SYSVOL will be. These measurements will lead into sizing both fixed disk and RAM allocation. Due to the (relatively) low cost of these components, the math does not need to be rigorous and precise. Content about how to evaluate this for both existing and new environments can be found in the Data Storage series of articles. Specifically, refer to the following articles:

- **For existing environments** – The section titled "To activate logging of disk space that is freed by defragmentation" in the article Storage Limits.
- **For new environments** – The article titled Growth Estimates for Active Directory Users and Organizational Units.

> **NOTE**
> The articles are based on data size estimates made at the time of the release of Active Directory in Windows 2000. Use object sizes that reflect the actual size of objects in your environment.

When reviewing existing environments with multiple domains, there may be variations in database sizes. Where this is true, use the smallest global catalog (GC) and non-GC sizes.

The database size can vary between operating system versions. DCs that run earlier operating systems such as Windows Server 2003 has a smaller database size than a DC that runs a later operating system such as Windows Server 2008 R2, especially when features such Active Directory Recycle Bin or Credential Roaming are enabled.

> **NOTE**
> - For new environments, notice that the estimates in Growth Estimates for Active Directory Users and Organizational Units indicate that 100,000 users (in the same domain) consume about 450 MB of space. Please note that the attributes populated can have a huge impact on the total amount. Attributes will be populated on many objects by both third-party and Microsoft products, including Microsoft Exchange Server and Lync. An evaluation based on the portfolio of the products in the environment is preferred, but the exercise of detailing out the math and testing for precise estimates for all but the largest environments may not actually be worth significant time and effort.
> - Ensure that 110% of the NTDS.dit size is available as free space in order to enable offline defrag, and plan for growth over a three to five year hardware lifespan. Given how cheap storage is, estimating storage at 300% the size of the DIT as storage allocation is safe to accommodate growth and the potential need for offline defrag.

Virtualization considerations for storage

In a scenario where multiple Virtual Hard Disk (VHD) files are being allocated on a single volume use a fixed sized disk of at least 210% (100% of the DIT + 110% free space) the size of the DIT to ensure that there is adequate space reserved.

Calculation summary example

DATA COLLECTED FROM EVALUATION PHASE	SIZE
NTDS.dit size	35 GB
Modifier to allow for offline defrag	2.1 GB
Total storage needed	73.5 GB

> **NOTE**
> This storage needed is in addition to the storage needed for SYSVOL, operating system, page file, temporary files, local cached data (such as installer files), and applications.

Storage performance

Evaluating performance of storage

As the slowest component within any computer, storage can have the biggest adverse impact on client

experience. For those environments large enough for which the RAM sizing recommendations are not feasible, the consequences of overlooking planning storage for performance can be devastating. Also, the complexities and varieties of storage technology further increase the risk of failure as the relevance of long standing best practices of "put operating system, logs, and database" on separate physical disks is limited in it's useful scenarios. This is because the long standing best practice is based on the assumption that is that a "disk" is a dedicated spindle and this allowed I/O to be isolated. This assumptions that make this true are no longer relevant with the introduction of:

- RAID
- New storage types and virtualized and shared storage scenarios
- Shared spindles on a Storage Area Network (SAN)
- VHD file on a SAN or network-attached storage
- Solid State Drives
- Tiered storage architectures (i.e. SSD storage tier caching larger spindle based storage)

Specifically, shared storage (RAID, SAN, NAS, JBOD (i.e. Storage Spaces), VHD) all have the ability to be oversubscribed/overloaded by other work loads that are placed on the back end storage. They also add in the challenge that SAN/network/driver issues (everything between the physical disk and the AD application) can cause throttling and/or delays. For clarification, these are not "bad" configurations, they are more complex configurations that require every component along the way to be working properly, thus requiring additional attention to ensure that performance is acceptable. See Appendix C, subsection "Introducing SANs," and Appendix D later in this document for more detailed explanations. Also, whereas Solid State Drives do not have the limitation of spinning disks (Hard Drives) regarding only allowing one IO at a time to be processed, they do still have IO limitations, and overloading/oversubscribing of SSDs is possible. In short, the end goal of all storage performance efforts, regardless of underlying storage architecture and design, is to ensure that the needed amount of Input/output Operations Per Second (IOPS) is available and that those IOPS happen within an acceptable time frame (as specified elsewhere in this document). For those scenarios with locally attached storage, reference Appendix C for the basics in how to design traditional local storage scenarios. These principals are generally applicable to more complex storage tiers and will also help in dialog with the vendors supporting backend storage solutions.

- Given the wide breadth of storage options available, it is recommended to engage the expertise of hardware support teams or vendors to ensure that the specific solution meets the needs of AD DS. The following numbers are the information that would be provided to the storage specialists.

For environments where the database is too large to be held in RAM, use the performance counters to determine how much I/O needs to be supported:

- LogicalDisk(*)\Avg Disk sec/Read (for example, if NTDS.dit is stored on the D:/ drive, the full path would be LogicalDisk(D:)\Avg Disk sec/Read)
- LogicalDisk(*)\Avg Disk sec/Write
- LogicalDisk(*)\Avg Disk sec/Transfer
- LogicalDisk(*)\Reads/sec
- LogicalDisk(*)\Writes/sec
- LogicalDisk(*)\Transfers/sec

These should be sampled in 15/30/60 minute intervals to benchmark the demands of the current environment.

Evaluating the results

NOTE

The focus is on reads from the database as this is usually the most demanding component, the same logic can be applied to writes to the log file by substituting LogicalDisk(*<NTDS Log>*)\Avg Disk sec/Write and LogicalDisk(*<NTDS Log>*)\Writes/sec):

- LogicalDisk(*<NTDS>*)\Avg Disk sec/Read indicates whether or not the current storage is adequately sized. If the results are roughly equal to the Disk Access Time for the disk type, LogicalDisk(*<NTDS>*)\Reads/sec is a valid measure. Check the manufacturer specifications for the storage on the back end, but good ranges for LogicalDisk(*<NTDS>*)\Avg Disk sec/Read would roughly be:
  - 7200 – 9 to 12.5 milliseconds (ms)
  - 10,000 – 6 to 10 ms
  - 15,000 – 4 to 6 ms
  - SSD – 1 to 3 ms
  - **NOTE**

    Recommendations exist stating that storage performance is degraded at 15ms to 20ms (depending on source). The difference between the above values and the other guidance is that the above values are the normal operating range. The other recommendations are troubleshooting guidance to identify when client experience significantly degrades and becomes noticeable. Reference Appendix C for a deeper explanation.
- LogicalDisk(*<NTDS>*)\Reads/sec is the amount of I/O that is being performed.
  - If LogicalDisk(*<NTDS>*)\Avg Disk sec/Read is within the optimal range for the backend storage, LogicalDisk(*<NTDS>*)\Reads/sec can be used directly to size the storage.
  - If LogicalDisk(*<NTDS>*)\Avg Disk sec/Read is not within the optimal range for the backend storage, additional I/O is needed according to the following formula: (LogicalDisk(*<NTDS>*)\Avg Disk sec/Read) ÷ (Physical Media Disk Access Time) × (LogicalDisk(*<NTDS>*)\Avg Disk sec/Read)

Considerations:

- Note that if the server is configured with a sub-optimal amount of RAM, these values will be inaccurate for planning purposes. They will be erroneously on the high side and can still be used as a worst case scenario.
- Adding/Optimizing RAM specifically will drive a decrease in the amount of read I/O (LogicalDisk(*<NTDS>*)\Reads/Sec. This means the storage solution may not have to be as robust as initially calculated. Unfortunately, anything more specific than this general statement is environmentally dependent on client load and general guidance cannot be provided. The best option is to adjust storage sizing after optimizing RAM.

Virtualization considerations for performance

Similar to all of the preceding virtualization discussions, the key here is to ensure that the underlying shared infrastructure can support the DC load plus the other resources using the underlying shared media and all pathways to it. This is true whether a physical domain controller is sharing the same underlying media on a SAN, NAS, or iSCSI infrastructure as other servers or applications, whether it is a guest using pass through access to a SAN, NAS, or iSCSI infrastructure that shares the underlying media, or if the guest is using a VHD file that resides on shared media locally or a SAN, NAS, or iSCSI infrastructure. The planning exercise is all about making sure that the underlying media can support the total load of all consumers.

Also, from a guest perspective, as there are additional code paths that must be traversed, there is a performance impact to having to go through a host to access any storage. Not surprisingly, storage performance testing indicates that the virtualizing has an impact on throughput that is subjective to the processor utilization of the host system (see Appendix A: CPU Sizing Criteria), which is obviously influenced by the resources of the host demanded by the guest. This contributes to the virtualization considerations regarding processing needs in a virtualized scenario (see Virtualization considerations for processing).

Making this more complex is that there are a variety of different storage options that are available that all have different performance impacts. As a safe estimate when migrating from physical to virtual, use a multiplier of 1.10 to adjust for different storage options for virtualized guests on Hyper-V, such as pass-through storage, SCSI Adapter, or IDE. The adjustments that need to be made when transferring between the different storage scenarios are irrelevant as to whether the storage is local, SAN, NAS, or iSCSI.

Calculation summary example

Determining the amount of I/O needed for a healthy system under normal operating conditions:

- LogicalDisk(*<NTDS Database Drive>*)\Transfers/sec during the peak period 15 minute period
- To determine the amount of I/O needed for storage where the capacity of the underlying storage is exceeded:

> *Needed IOPS* = (LogicalDisk(*<NTDS Database Drive>*)\Avg Disk sec/Read ÷ *<Target Avg Disk sec/Read>*) × LogicalDisk(*<NTDS Database Drive>*)\Read/sec

COUNTER	VALUE
Actual LogicalDisk(*<NTDS Database Drive>*)\Avg Disk sec/Transfer	.02 seconds (20 milliseconds)
Target LogicalDisk(*<NTDS Database Drive>*)\Avg Disk sec/Transfer	.01 seconds
Multiplier for change in available IO	0.02 ÷ 0.01 = 2

VALUE NAME	VALUE
LogicalDisk(*<NTDS Database Drive>*)\Transfers/sec	400
Multiplier for change in available IO	2
Total IOPS needed during peak period	800

To determine the rate at which the cache is desired to be warmed:

- Determine the maximum acceptable time to warm the cache. It is either the amount of time that it should take to load the entire database from disk, or for scenarios where the entire database cannot be loaded in RAM, this would be the maximum time to fill RAM.
- Determine the size of the database, excluding white space. For more information, see Evaluating for storage.
- Divide the database size by 8 KB; that will be the total IOs necessary to load the database.
- Divide the total IOs by the number of seconds in the defined time frame.

Note that the rate calculated, while accurate, will not be exact because previously loaded pages are evicted if ESE is not configured to have a fixed cache size, and AD DS by default uses variable cache size.

DATA POINTS TO COLLECT	VALUES
Maximum acceptable time to warm	10 minutes (600 seconds)
Database size	2 GB

CALCULATION STEP	FORMULA	RESULT
Calculate size of database in pages	(2 GB × 1024 × 1024) = *Size of database in KB*	2,097,152 KB
Calculate number of pages in database	2,097,152 KB ÷ 8 KB = *Number of pages*	262,144 pages
Calculate IOPS necessary to fully warm the cache	262,144 pages ÷ 600 seconds = *IOPS needed*	437 IOPS

Processing

Evaluating Active Directory processor usage

For most environments, after storage, RAM, and networking are properly tuned as described in the Planning section, managing the amount of processing capacity will be the component that deserves the most attention. There are two challenges in evaluating CPU capacity needed:

- Whether or not the applications in the environment are being well-behaved in a shared services infrastructure, and is discussed in the section titled "Tracking Expensive and Inefficient Searches" in the article Creating More Efficient Microsoft Active Directory-Enabled Applications or migrating away from down-level SAM calls to LDAP calls.

  In larger environments, the reason this is important is that poorly coded applications can drive volatility in CPU load, "steal" an inordinate amount of CPU time from other applications, artificially drive up capacity needs, and unevenly distribute load against the DCs.

- As AD DS is a distributed environment with a large variety of potential clients, estimating the expense of a "single client" is environmentally subjective due to usage patterns and the type or quantity of applications leveraging AD DS. In short, much like the networking section, for broad applicability, this is better approached from the perspective of evaluating the total capacity needed in the environment.

For existing environments, as storage sizing was discussed previously, the assumption is made that storage is now properly sized and thus the data regarding processor load is valid. To reiterate, it is critical to ensure that the bottleneck in the system is not the performance of the storage. When a bottleneck exists and the processor is waiting, there are idle states that will go away once the bottleneck is removed. As processor wait states are removed, by definition, CPU utilization increases as it no longer has to wait on the data. Thus, collect performance counters "Logical Disk(*<NTDS Database Drive>*)\Avg Disk sec/Read" and "Process(lsass)\% Processor Time". The data in "Process(lsass)\% Processor Time" will be artificially low if "Logical Disk(*<NTDS Database Drive>*)\Avg Disk sec/Read" exceeds 10 to 15 ms, which is a general threshold that Microsoft support uses for troubleshooting storage-related performance issues. As before, it is recommended that sample intervals be either 15, 30, or 60 minutes. Anything less will generally be too volatile for good measurements; anything greater will smooth out daily peeks excessively.

Introduction

In order to plan capacity planning for domain controllers, processing power requires the most attention and understanding. When sizing systems to ensure maximum performance, there is always a component that is the bottleneck and in a properly sized Domain Controller this will be the processor.

Similar to the networking section where the demand of the environment is reviewed on a site-by-site basis, the same must be done for the compute capacity demanded. Unlike the networking section, where the available networking technologies far exceed the normal demand, pay more attention to sizing CPU capacity. As any environment of even moderate size; anything over a few thousand concurrent users can put significant load on the CPU.

Unfortunately, due to the huge variability of client applications that leverage AD, a general estimate of users per CPU is woefully inapplicable to all environments. Specifically, the compute demands are subject to user behavior and application profile. Therefore, each environment needs to be individually sized.

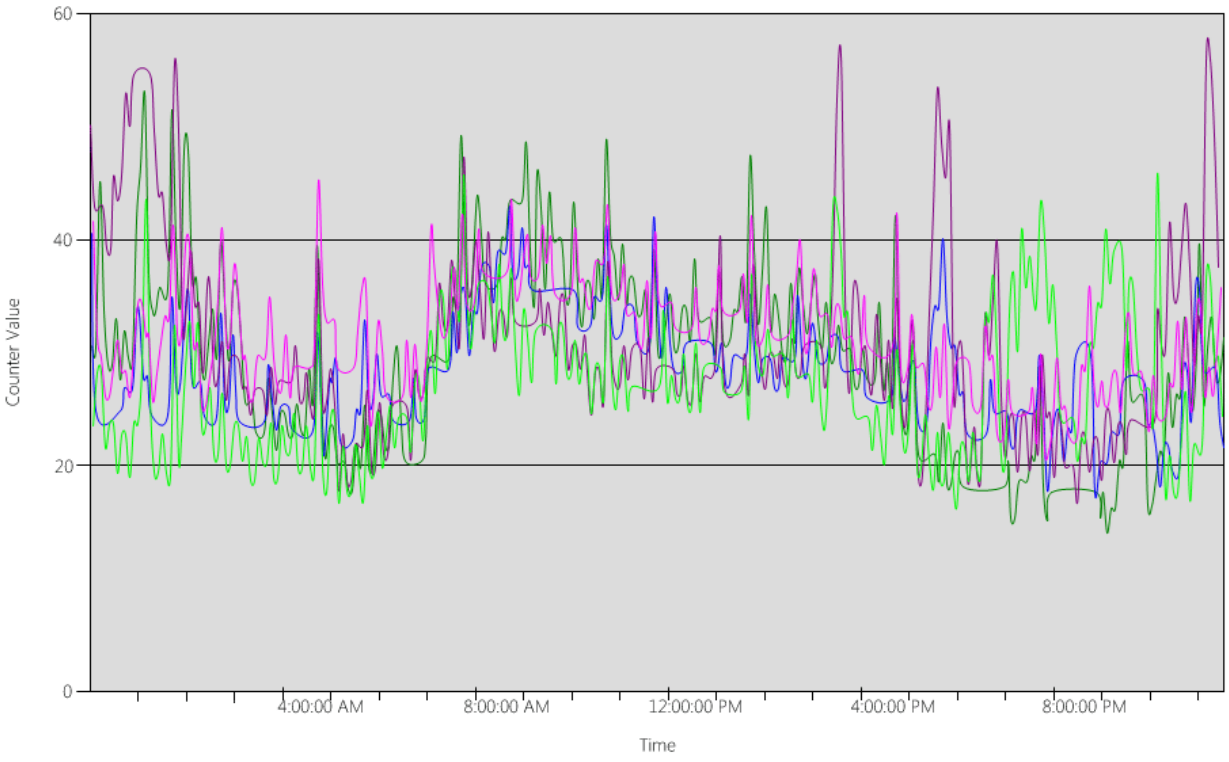
Target site behavior profile

As mentioned previously, when planning capacity for an entire site, the goal is to target a design with an *N* + 1 capacity design, such that failure of one system during the peak period will allow for continuation of service at a reasonable level of quality. That means that in an "*N*" scenario, load across all the boxes should be less than 100% (better yet, less than 80%) during the peak periods.

Additionally, if the applications and clients in the site are using best practices for locating domain controllers (that is, using the DsGetDcName function), the clients should be relatively evenly distributed with minor transient spikes due to any number of factors.

In the next example, the following assumptions are made:

- Each of the five DCs in the site has four of CPUs.
- Total target CPU usage during business hours is 40% under normal operating conditions ("*N* + 1") and 60% otherwise ("*N*"). During non-business hours, the target CPU usage is 80% because backup software and other maintenance are expected to consume all available resources.



Analyzing the data in the chart `(Processor Information(_Total)\% Processor Utility)` for each of the DCs:

- For the most part, the load is relatively evenly distributed which is what would be expected when clients use DC locator and have well written searches.
- There are a number of five-minute spikes of 10%, with some as large as 20%. Generally, unless they cause the capacity plan target to be exceeded, investigating these is not worthwhile.
- The peak period for all systems is between about 8:00 AM and 9:15 AM. With the smooth transition from about 5:00 AM through about 5:00 PM, this is generally indicative of the business cycle. The more randomized spikes of CPU usage on a box-by-box scenario between 5:00 PM and 4:00 AM would be outside of the capacity planning concerns.

> **NOTE**
>
> On a well-managed system, said spikes are might be backup software running, full system antivirus scans, hardware or software inventory, software or patch deployment, and so on. Because they fall outside the peak user business cycle, the targets are not exceeded.

- As each system is about 40% and all systems have the same numbers of CPUs, should one fail or be taken offline, the remaining systems would run at an estimated 53% (System D's 40% load is evenly split and added to System A's and System C's existing 40% load). For a number of reasons, this linear assumption is NOT perfectly accurate, but provides enough accuracy to gauge.

  Alternate scenario – Two domain controllers running at 40%: One domain controller fails, estimated CPU on the remaining one would be an estimated 80%. This far exceeds the thresholds outlined above for capacity plan and also starts to severely limit the amount of head room for the 10% to 20% seen in the load profile above, which means that the spikes would drive the DC to 90% to 100% during the "*N*" scenario and definitely degrade responsiveness.

Calculating CPU demands

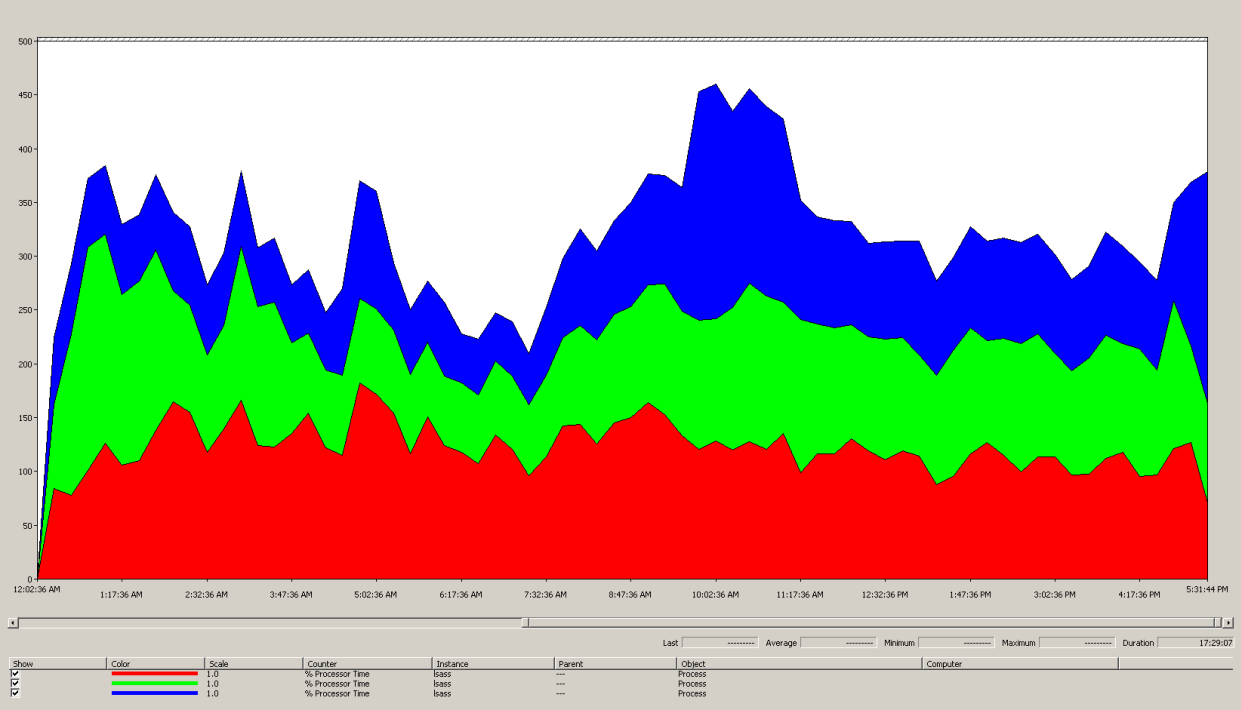
The "Process\% Processor Time" performance object counter sums the total amount of time that all of the threads of an application spend on the CPU and divides by the total amount of system time that has passed. The effect of this is that a multi-threaded application on a multi-CPU system can exceed 100% CPU time, and would be interpreted VERY differently than "Processor Information\% Processor Utility". In practice the "Process(lsass)\% Processor Time" can be viewed as the count of CPUs running at 100% that are necessary to support the process's demands. A value of 200% means that 2 CPUs, each at 100%, are needed to support the full AD DS load. Although a CPU running at 100% capacity is the most cost efficient from the perspective of money spent on CPUs and power and energy consumption, for a number of reasons detailed in Appendix A, better responsiveness on a multi-threaded system occurs when the system is not running at 100%.

To accommodate transient spikes in client load, it is recommended to target a peak period CPU of between 40% and 60% of system capacity. Working with the example above, that would mean that between 3.33 (60% target) and 5 (40% target) CPUs would be needed for the AD DS (lsass process) load. Additional capacity should be added in according to the demands of the base operating system and other agents required (such as antivirus, backup, monitoring, and so on). Although the impact of agents needs to be evaluated on a per environment basis, an estimate of between 5% and 10% of a single CPU can be made. In the current example, this would suggest that between 3.43 (60% target) and 5.1 (40% target) CPUs are necessary during peak periods.

The easiest way to do this is to use the "Stacked Area" view in Windows Reliability and Performance Monitor (perfmon), making sure all of the counters are scaled the same.

Assumptions:

- Goal is to reduce footprint to as few servers as possible. Ideally, one server would carry the load and an additional server added for redundancy (*N* + 1 scenario).



Knowledge gained from the data in the chart (Process(lsass)\% Processor Time):

- The business day starts ramping up around 7:00 and decreases at 5:00 PM.
- The peak busiest period is from 9:30 AM to 11:00 AM.

> **NOTE**
>
> All performance data is historical. The peak data point at 9:15 indicates the load from 9:00 to 9:15.

- There are spikes before 7:00 AM which could indicate either load from different time zones or background infrastructure activity, such as backups. Because the peak at 9:30 AM exceeds this activity, it is not relevant.
- There are three domain controllers in the site.

At maximum load, lsass consumes about 485% of one CPU, or 4.85 CPUs running at 100%. As per the math earlier, this means the site needs about 12.25 CPUs for AD DS. Add in the above suggestions of 5% to 10% for background processes and that means replacing the server today would need approximately 12.30 to 12.35 CPUs to support the same load. An environmental estimate for growth now needs to be factored in.

When to tune LDAP weights

There are several scenarios where tuning LdapSrvWeight should be considered. Within the context of capacity planning, this would be done when the application or user loads are not evenly balanced, or the underlying systems are not evenly balanced in terms of capability. Reasons to do so beyond capacity planning are outside of the scope of this article.

There are two common reasons to tune LDAP Weights:

- The PDC emulator is an example that affects every environment for which user or application load behavior is not evenly distributed. As certain tools and actions target the PDC emulator, such as the Group Policy management tools, second attempts in the case of authentication failures, trust establishment, and so on, CPU resources on the PDC emulator may be more heavily demanded than elsewhere in the site.
  - It is only useful to tune this if there is a noticeable difference in CPU utilization in order to reduce the load on the PDC emulator and increase the load on other domain controllers will allow a more even distribution of load.
  - In this case, set LDAPSrvWeight between 50 and 75 for the PDC emulator.
- Servers with differing counts of CPUs (and speeds) in a site. For example, say there are two eight-core

servers and one four-core server. The last server has half the processors of the other two servers. This means that a well distributed client load will increase the average CPU load on the four-core box to roughly twice that of the eight-core boxes.

- For example, the two eight-core boxes would be running at 40% and the four-core box would be running at 80%.
- Also, consider the impact of loss of one eight-core box in this scenario, specifically the fact that the four-core box would now be overloaded.

Example 1 - PDC

SYSTEM	UTILIZATION WITH DEFAULTS	NEW LDAPSRVWEIGHT	ESTIMATED NEW UTILIZATION
DC 1 (PDC emulator)	53%	57	40%
DC 2	33%	100	40%
DC 3	33%	100	40%

The catch here is that if the PDC emulator role is transferred or seized, particularly to another domain controller in the site, there will be a dramatic increase on the new PDC emulator.

Using the example from the section Target site behavior profile, an assumption was made that all three domain controllers in the site had four CPUs. What should happen, under normal conditions, if one of the domain controllers had eight CPUs? There would be two domain controllers at 40% utilization and one at 20% utilization. While this is not bad, there is an opportunity to balance the load a little bit better. Leverage LDAP weights to accomplish this. An example scenario would be:

Example 2 - Differing CPU counts

SYSTEM	PROCESSOR INFORMATION\ % PROCESSOR UTILITY(_TOTAL) UTILIZATION WITH DEFAULTS	NEW LDAPSRVWEIGHT	ESTIMATED NEW UTILIZATION
4-CPU DC 1	40	100	30%
4-CPU DC 2	40	100	30%
8-CPU DC 3	20	200	30%

Be very careful with these scenarios though. As can be seen above, the math looks really nice and pretty on paper. But throughout this article, planning for an "*N* + 1" scenario is of paramount importance. The impact of one DC going offline must be calculated for every scenario. In the immediately preceding scenario where the load distribution is even, in order to ensure a 60% load during an "*N*" scenario, with the load balanced evenly across all servers, the distribution will be fine as the ratios stay consistent. Looking at the PDC emulator tuning scenario, and in general any scenario where user or application load is unbalanced, the effect is very different:

SYSTEM	TUNED UTILIZATION	NEW LDAPSRVWEIGHT	ESTIMATED NEW UTILIZATION
DC 1 (PDC emulator)	40%	85	47%
DC 2	40%	100	53%

SYSTEM	TUNED UTILIZATION	NEW LDAPSRVWEIGHT	ESTIMATED NEW UTILIZATION
DC 3	40%	100	53%

Virtualization considerations for processing

There are two layers of capacity planning that need to be done in a virtualized environment. At the host level, similar to the identification of the business cycle outlined for the domain controller processing previously, thresholds during the peak period need to be identified. Because the underlying principals are the same for a host machine scheduling guest threads on the CPU as for getting AD DS threads on the CPU on a physical machine, the same goal of 40% to 60% on the underlying host are recommended. At the next layer, the guest layer, since the principals of thread scheduling have not changed, the goal within the guest remains in the 40% to 60% range.

In a direct mapped scenario, one guest per host, all the capacity planning done to this point needs to be added to the requirements (RAM, disk, network) of the underlying host operating system. In a shared host scenario, testing indicates that there is 10% impact on the efficiency of the underlying processors. That means if a site needs 10 CPUs at a target of 40%, the recommended amount of virtual CPUs to allocate across all the "*N*" guests would be 11. In a site with a mixed distribution of physical servers and virtual servers, the modifier only applies to the VMs. For example, if a site has an "*N* + 1" scenario, one physical or direct-mapped server with 10 CPUs would be about equivalent to one guest with 11 CPUs on a host, with 11 CPUs reserved for the domain controller.

Throughout the analysis and calculation of the CPU quantities necessary to support AD DS load, the numbers of CPUs that map to what can be purchased in terms physical hardware do not necessarily map cleanly. Virtualization eliminates the need to round up. Virtualization decreases the effort necessary to add compute capacity to a site, given the ease with which a CPU can be added to a VM. It does not eliminate the need to accurately evaluate the compute power needed so that the underlying hardware is available when additional CPUs need to be added to the guests. As always, remember to plan and monitor for growth in demand.

Calculation summary example

SYSTEM	PEAK CPU
DC 1	120%
DC 2	147%
Dc 3	218%
Total CPU being used	485%

TARGET SYSTEM(S) COUNT	TOTAL BANDWIDTH (FROM ABOVE)
CPUs needed at 40% target	4.85 ÷ .4 = 12.25

Repeating due to the importance of this point, *remember to plan for growth*. Assuming 50% growth over the next three years, this environment will need 18.375 CPUs (12.25 × 1.5) at the three-year mark. An alternate plan would be to review after the first year and add in additional capacity as needed.

Cross-trust client authentication load for NTLM

Evaluating cross-trust client authentication load

Many environments may have one or more domains connected by a trust. An authentication request for an identity in another domain that does not use Kerberos authentication needs to traverse a trust using the domain

controller's secure channel to another domain controller either in the destination domain or the next domain in the path to the destination domain. The number of concurrent calls using the secure channel that a domain controller can make to a domain controller in a trusted domain is controlled by a setting known as **MaxConcurrentAPI**. For domain controllers, ensuring that the secure channel can handle the amount of load is accomplished by one of two approaches: tuning **MaxConcurrentAPI** or, within a forest, creating shortcut trusts. To gauge the volume of traffic across an individual trust, refer to How to do performance tuning for NTLM authentication by using the MaxConcurrentApi setting.

During data collection, this, as with all the other scenarios, must be collected during the peak busy periods of the day for the data to be useful.

> **NOTE**
> Intraforest and interforest scenarios may cause the authentication to traverse multiple trusts and each stage would need to be tuned.

Planning

There are a number of applications that use NTLM authentication by default, or use it in a certain configuration scenario. Application servers grow in capacity and service an increasing number of active clients. There is also a trend that clients keep sessions open for a limited time and rather reconnect on a regular basis (such as email pull sync). Another common example for high NTLM load is web proxy servers that require authentication for Internet access.

These applications can cause a significant load for NTLM authentication, which can put significant stress on the DCs, especially when users and resources are in different domains.

There are multiple approaches to managing cross-trust load, which in practice are used in conjunction rather than in an exclusive either/or scenario. The possible options are:

- Reduce cross-trust client authentication by locating the services that a user consumes in the same domain that the user is resident in.
- Increase the number of secure-channels available. This is relevant to intraforest and cross-forest traffic and are known as shortcut trusts.
- Tune the default settings for **MaxConcurrentAPI**.

For tuning **MaxConcurrentAPI** on an existing server, the equation is:

> *New_MaxConcurrentApi_setting* ≥ (*semaphore_acquires* + *semaphore_time-outs*) × *average_semaphore_hold_time* ÷ *time_collection_length*

For more information, see KB article 2688798: How to do performance tuning for NTLM authentication by using the MaxConcurrentApi setting.

Virtualization considerations

None, this is an operating system tuning setting.

Calculation summary example

DATA TYPE	VALUE
Semaphore Acquires (Minimum)	6,161
Semaphore Acquires (Maximum)	6,762

DATA TYPE	VALUE
Semaphore Timeouts	0
Average Semaphore Hold Time	0.012
Collection Duration (seconds)	1:11 minutes (71 seconds)
Formula (from KB 2688798)	((6762 – 6161) + 0) × 0.012 /
Minimum value for **MaxConcurrentAPI**	((6762 – 6161) + 0) × 0.012 ÷ 71 = .101

For this system for this time period, the default values are acceptable.

Monitoring for compliance with capacity planning goals

Throughout this article, it has been discussed that planning and scaling go towards utilization targets. Here is a summary chart of the recommended thresholds that must be monitored to ensure the systems are operating within adequate capacity thresholds. Keep in mind that these are not performance thresholds, but capacity planning thresholds. A server operating in excess of these thresholds will work, but is time to start validating that all the applications are well behaved. If said applications are well behaved, it is time to start evaluating hardware upgrades or other configuration changes.

CATEGORY	PERFORMANCE COUNTER	INTERVAL/SAMPLING	TARGET	WARNING
Processor	Processor Information(_Total)\% Processor Utility	60 min	40%	60%
RAM (Windows Server 2008 R2 or earlier)	Memory\Available MB	< 100 MB	N/A	< 100 MB
RAM (Windows Server 2012)	Memory\Long-Term Average Standby Cache Lifetime(s)	30 min	Must be tested	Must be tested
Network	Network Interface(*)\Bytes Sent/sec Network Interface(*)\Bytes Received/sec	30 min	40%	60%
Storage	LogicalDisk(*<NTDS Database Drive>*)\Avg Disk sec/Read LogicalDisk(*<NTDS Database Drive>*)\Avg Disk sec/Write	60 min	10 ms	15 ms

CATEGORY	PERFORMANCE COUNTER	INTERVAL/SAMPLING	TARGET	WARNING
AD Services	Netlogon(*)\Average Semaphore Hold Time	60 min	0	1 second

Appendix A: CPU sizing criteria

Definitions

Processor (microprocessor) – a component that reads and executes program instructions

CPU – Central Processing Unit

Multi-Core processor – multiple CPUs on the same integrated circuit

Multi-CPU – multiple CPUs, not on the same integrated circuit

Logical Processor – one logical computing engine from the perspective of the operating system

This includes hyper-threaded, one core on multi-core processor, or a single core processor.

As today's server systems have multiple processors, multiple multi-core processors, and hyper-threading, this information is generalized to cover both scenarios. As such, the term logical processor will be used as it represents the operating system and application perspective of the available computing engines.

Thread-level parallelism

Each thread is an independent task, as each thread has its own stack and instructions. Because AD DS is multi-threaded and the number of available threads can be tuned by using How to view and set LDAP policy in Active Directory by using Ntdsutil.exe, it scales well across multiple logical processors.

Data-level parallelism

This involves sharing data across multiple threads within one process (in the case of the AD DS process alone) and across multiple threads in multiple processes (in general). With concern to over-simplifying the case, this means that any changes to data are reflected to all running threads in all the various levels of cache (L1, L2, L3) across all cores running said threads as well as updating shared memory. Performance can degrade during write operations while all the various memory locations are brought consistent before instruction processing can continue.

CPU speed vs. multiple-core considerations

The general rule of thumb is faster logical processors reduce the duration it takes to process a series of instructions, while more logical processors means that more tasks can be run at the same time. These rules of thumb break down as the scenarios become inherently more complex with considerations of fetching data from shared-memory, waiting on data-level parallelism, and the overhead of managing multiple threads. This is also why scalability in multi-core systems is not linear.

Consider the following analogies in these considerations: think of a highway, with each thread being an individual car, each lane being a core, and the speed limit being the clock speed.

1. If there is only one car on the highway, it doesn't matter if there are two lanes or 12 lanes. That car is only going to go as fast as the speed limit will allow.
2. Assume that the data the thread needs is not immediately available. The analogy would be that a segment of road is shutdown. If there is only one car on the highway, it doesn't matter what the speed limit is until the lane is reopened (data is fetched from memory).
3. As the number of cars increase, the overhead to manage the number of cars increases. Compare the experience of driving and the amount of attention necessary when the road is practically empty (such as late

evening) versus when the traffic is heavy (such as mid-afternoon, but not rush hour). Also, consider the amount of attention necessary when driving on a two-lane highway, where there is only one other lane to worry about what the drivers are doing, versus a six-lane highway where one has to worry about what a lot of other drivers are doing.

> **NOTE**
> The analogy about the rush hour scenario is extended in the next section: Response Time/How the System Busyness Impacts Performance.

As a result, specifics about more or faster processors become highly subjective to application behavior, which in the case of AD DS is very environmentally specific and even varies from server to server within an environment. This is why the references earlier in the article do not invest heavily in being overly precise, and a margin of safety is included in the calculations. When making budget-driven purchasing decisions, it is recommended that optimizing usage of the processors at 40% (or the desired number for the environment) occurs first, before considering buying faster processors. The increased synchronization across more processors reduces the true benefit of more processors from the linear progression (2× the number of processors provides less than 2× available additional compute power).

> **NOTE**
> Amdahl's Law and Gustafson's Law are the relevant concepts here.

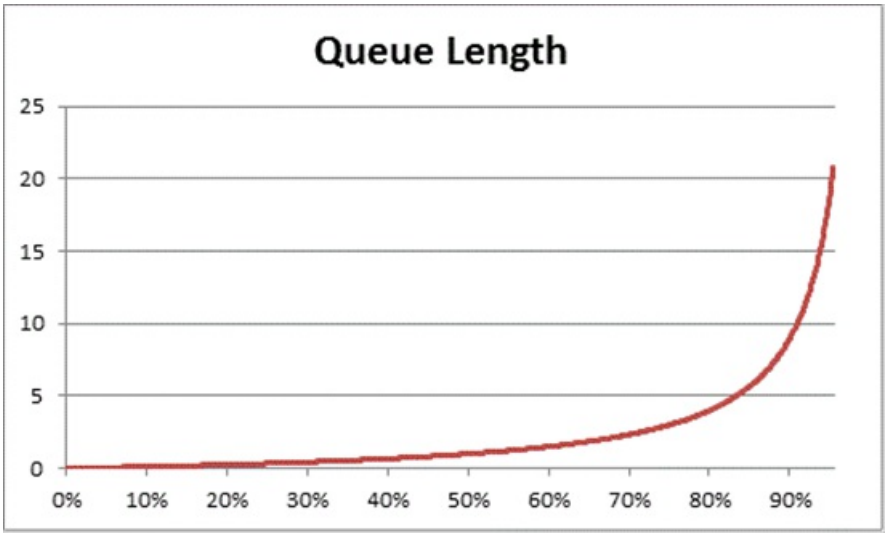
Response time/How the system busyness impacts performance

Queuing theory is the mathematical study of waiting lines (queues). In queuing theory, the Utilization Law is represented by the equation:

$U\text{k} = B \div T$

Where $U\text{k}$ is the utilization percentage, B is the amount of time busy, and T is the total time the system was observed. Translated into the context of Windows, this means the number of 100-nanosecond (ns) interval threads that are in a Running state divided by how many 100-ns intervals were available in given time interval. This is exactly the formula for calculating % Processor Utility (reference Processor Object and PERF_100NSEC_TIMER_INV).

Queuing theory also provides the formula: $N = U\text{k} \div (1 - U\text{k})$ to estimate the number of waiting items based on utilization (N is the length of the queue). Charting this over all utilization intervals provides the following estimates to how long the queue to get on the processor is at any given CPU load.



It is observed that after 50% CPU load, on average there is always a wait of one other item in the queue, with a

noticeably rapid increase after about 70% CPU utilization.

Returning to the driving analogy used earlier in this section:

- The busy times of "mid-afternoon" would, hypothetically, fall somewhere into the 40% to 70% range. There is enough traffic such that one's ability to pick any lane is not majorly restricted, and the chance of another driver being in the way, while high, does not require the level of effort to "find" a safe gap between other cars on the road.
- One will notice that as traffic approaches rush hour, the road system approaches 100% capacity. Changing lanes can become very challenging because cars are so close together that increased caution must be exercised to do so.

This is why the long term averages for capacity conservatively estimated at 40% allows for head room for abnormal spikes in load, whether said spikes transitory (such as poorly coded queries that run for a few minutes) or abnormal bursts in general load (the morning of the first day after a long weekend).

The above statement regards % Processor Time calculation being the same as the Utilization Law is a bit of a simplification for the ease of the general reader. For those more mathematically rigorous:

- Translating the PERF_100NSEC_TIMER_INV
  - B = The number of 100-ns intervals "Idle" thread spends on the logical processor. The change in the "X" variable in the PERF_100NSEC_TIMER_INV calculation
  - T = the total number of 100-ns intervals in a given time range. The change in the "Y" variable in the PERF_100NSEC_TIMER_INV calculation.
  - U_k = The utilization percentage of the logical processor by the "Idle Thread" or % Idle Time.
- Working out the math:
  - U_k = 1 – %Processor Time
  - %Processor Time = 1 – U_k
  - %Processor Time = 1 – B / T
  - %Processor Time = 1 – $X1 – X0 / Y1 – Y0$

Applying the concepts to capacity planning

The preceding math may make determinations about the number of logical processors needed in a system seem overwhelmingly complex. This is why the approach to sizing the systems is focused on determining maximum target utilization based on current load and calculating the number of logical processors required to get there. Additionally, while logical processor speeds will have a significant impact on performance, cache efficiencies, memory coherence requirements, thread scheduling and synchronization, and imperfectly balanced client load will all have significant impacts on performance that will vary on a server-by-server basis. With the relatively cheap cost of compute power, attempting to analyze and determine the perfect number of CPUs needed becomes more an academic exercise than it does provide business value.

Forty percent is not a hard and fast requirement, it is a reasonable start. Various consumers of Active Directory require various levels of responsiveness. There may be scenarios where environments can run at 80% or 90% utilization as a sustained average, as the increased wait times for access to the processor will not noticeably impact client performance. It is important to re-iterate that there are many areas in the system that are much slower than the logical processor in the system, including access to RAM, access to disk, and transmitting the response over the network. All of these items need to be tuned in conjunction. Examples:

- Adding more processors to a system running 90% that is disk-bound is probably not going to significantly improve performance. Deeper analysis of the system will probably identify that there are a lot of threads that are not even getting on the processor because they are waiting on I/O to complete.
- Resolving the disk-bound issues potentially means that threads that were previously spending a lot of time in a waiting state will no longer be in a waiting state for I/O and there will be more competition for CPU time, meaning that the 90% utilization in the previous example will go to 100% (because it can not go higher).

Both components need to be tuned in conjunction.

> **NOTE**
>
> Processor Information(*)\% Processor Utility can exceed 100% with systems that have a "Turbo" mode. This is where the CPU exceeds the rated processor speed for short periods. Reference CPU manufacturers documentation and description of the counter for greater insight.

Discussing whole system utilization considerations also brings into the conversation domain controllers as virtualized guests. Response time/How the system busyness impacts performance applies to both the host and the guest in a virtualized scenario. This is why in a host with only one guest, a domain controller (and generally any system) has near the same performance it does on physical hardware. Adding additional guests to the hosts increases the utilization of the underlying host, thereby increasing the wait times to get access to the processors as explained previously. In short, logical processor utilization needs to be managed at both the host and at the guest levels.

Extending the previous analogies, leaving the highway as the physical hardware, the guest VM will be analogized with a bus (an express bus that goes straight to the destination the rider wants). Imagine the following four scenarios:

- It is off hours, a rider gets on a bus that is nearly empty, and the bus gets on a road that is also nearly empty. As there is no traffic to contend with, the rider has a nice easy ride and gets there just as fast as if the rider had driven instead. The rider's travel times are still constrained by the speed limit.
- It is off hours so the bus is nearly empty but most of the lanes on the road are closed, so the highway is still congested. The rider is on an almost-empty bus on a congested road. While the rider does not have a lot of competition in the bus for where to sit, the total trip time is still dictated by the rest of the traffic outside.
- It is rush hour so the highway and the bus are congested. Not only does the trip take longer, but getting on and off the bus is a nightmare because people are shoulder to shoulder and the highway is not much better. Adding more buses (logical processors to the guest) does not mean they can fit on the road any more easily, or that the trip will be shortened.
- The final scenario, though it may be stretching the analogy a little, is where the bus is full, but the road is not congested. While the rider will still have trouble getting on and off the bus, the trip will be efficient after the bus is on the road. This is the only scenario where adding more buses (logical processors to the guest) will improve guest performance.

From there it is relatively easy to extrapolate that there are a number of scenarios in between the 0%-utilized and the 100%-utilized state of the road and the 0%- and 100%-utilized state of the bus that have varying degrees of impact.

Applying the principals above of 40% CPU as reasonable target for the host as well as the guest is a reasonable start for the same reasoning as above, the amount of queuing.

Appendix B: Considerations regarding different processor speeds, and the effect of processor power management on processor speeds

Throughout the sections on processor selection the assumption is made that the processor is running at 100% of clock speed the entire time the data is being collected and that the replacement systems will have the same speed processors. Despite both assumptions in practice being false, particularly with Windows Server 2008 R2 and later, where the default power plan is **Balanced**, the methodology still stands as it is the conservative approach. While the potential error rate may increase, it only increases the margin of safety as processor speeds increase.

- For example, in a scenario where 11.25 CPUs are demanded, if the processors were running at half speed when the data was collected, the more accurate estimate might be 5.125 ÷ 2.

- It is impossible to guarantee that doubling the clock speeds would double the amount of processing that happens for a given time period. This is due to the fact the amount of time that the processors spend waiting on RAM or other system components could stay the same. The net effect is that the faster processors might spend a greater percentage of the time idle while waiting on data to be fetched. Again, it is recommended to stick with the lowest common denominator, being conservative, and avoid trying to calculate a potentially false level of accuracy by assuming a linear comparison between processor speeds.

Alternatively, if processor speeds in replacement hardware are lower than current hardware, it would be safe to increase the estimate of processors needed by a proportionate amount. For example, it is calculated that 10 processors are needed to sustain the load in a site, and the current processors are running at 3.3 Ghz and replacement processors will run at 2.6 Ghz, this is a 21% decrease in speed. In this case, 12 processors would be the recommended amount.

That said, this variability would not change the Capacity Management processor utilization targets. As processor clock speeds will be adjusted dynamically based on the load demanded, running the system under higher loads will generate a scenario where the CPU spends more time in a higher clock speed state, making the ultimate goal to be at 40% utilization in a 100% clock speed state at peak. Anything less than that will generate power savings as CPU speeds will be throttled back during off peak scenarios.

> **NOTE**
> An option would be to turn off power management on the processors (setting the power plan to **High Performance**) while data is collected. That would give a more accurate representation of the CPU consumption on the target server.

To adjust estimates for different processors, it used to be safe, excluding other system bottlenecks outlined above, to assume that doubling processor speeds doubled the amount of processing that could be performed. Today, the internal architecture of processors is different enough between processors, that a safer way to gauge the effects of using different processors than data was taken from is to leverage the SPECint_rate2006 benchmark from Standard Performance Evaluation Corporation.

1. Find the SPECint_rate2006 scores for the processor that are in use and that plan to be used.
    a. On the website of the Standard Performance Evaluation Corporation, select **Results**, highlight **CPU2006**, and select **Search all SPECint_rate2006 results**.
    b. Under **Simple Request**, enter the search criteria for the target processor, for example **Processor Matches E5-2630 (baselinetarget)** and **Processor Matches E5-2650 (baseline)**.
    c. Find the server and processor configuration to be used (or something close, if an exact match is not available) and note the value in the **Result** and **# Cores** columns.
2. To determine the modifier use the following equation:

    ```
    ((Target platform per-core score value) × (MHz per-core of baseline platform)) ÷ ((Baseline per-core score value) × (MHz per-core of target platform))
    ```

    Using the above example:

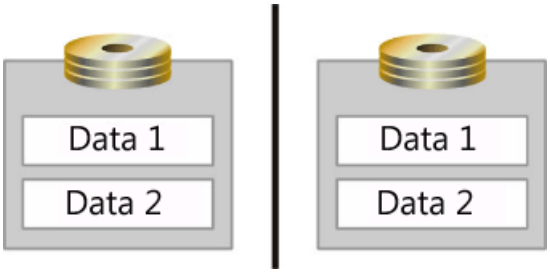
    ```
    (35.83 × 2000) ÷ (33.75 × 2300) = 0.92
    ```

3. Multiply the estimated number of processors by the modifier. In the above case to go from the E5-2650 processor to the E5-2630 processor multiply the calculated 11.25 CPUs × 0.92 = 10.35 processors needed.

Appendix C: Fundamentals regarding the operating system

interacting with storage

The queuing theory concepts outlined in Response time/How the system busyness impacts performance are also applicable to storage. Having a familiarity of how the operating system handles I/O is necessary to apply these concepts. In the Microsoft Windows operating system, a queue to hold the I/O requests is created for each physical disk. However, a clarification on physical disk needs to be made. Array controllers and SANs present aggregations of spindles to the operating system as single physical disks. Additionally, array controllers and SANs can aggregate multiple disks into one array set and then split this array set into multiple "partitions", which is in turn presented to the operating system as multiple physical disks (ref. figure).



In this figure the two spindles are mirrored and split into logical areas for data storage (Data 1 and Data 2). These logical areas are viewed by the operating system as separate physical disks.

Although this can be highly confusing, the following terminology is used throughout this appendix to identify the different entities:

- **Spindle** – the device that is physically installed in the server.
- **Array** – a collection of spindles aggregated by controller.
- **Array partition** – a partitioning of the aggregated array
- **LUN** – an array, used when referring to SANs
- **Disk** – What the operating system observes to be a single physical disk.
- **Partition** – a logical partitioning of what the operating system perceives as a physical disk.

Operating system architecture considerations

The operating system creates a First In/First Out (FIFO) I/O queue for each disk that is observed; this disk may be representing a spindle, an array, or an array partition. From the operating system perspective, with regard to handling I/O, the more active queues the better. As a FIFO queue is serialized, meaning that all I/Os issued to the storage subsystem must be processed in the order the request arrived. By correlating each disk observed by the operating system with a spindle/array, the operating system now maintains an I/O queue for each unique set of disks, thereby eliminating contention for scarce I/O resources across disks and isolating I/O demand to a single disk. As an exception, Windows Server 2008 introduces the concept of I/O prioritization, and applications designed to use the "Low" priority fall out of this normal order and take a back seat. Applications not specifically coded to leverage the "Low" priority default to "Normal."

Introducing simple storage subsystems

Starting with a simple example (a single hard drive inside a computer) a component-by-component analysis will be given. Breaking this down into the major storage subsystem components, the system consists of:

- **1** – 10,000 RPM Ultra Fast SCSI HD (Ultra Fast SCSI has a 20 MB/s transfer rate)
- **1** – SCSI Bus (the cable)
- **1** – Ultra Fast SCSI Adapter
- **1** – 32-bit 33 MHz PCI bus

Once the components are identified, an idea of how much data can transit the system, or how much I/O can be handled, can be calculated. Note that the amount of I/O and quantity of data that can transit the system is correlated, but not the same. This correlation depends on whether the disk I/O is random or sequential and the

block size. (All data is written to the disk as a block, but different applications using different block sizes.) On a component-by-component basis:

- **The hard drive** – The average 10,000-RPM hard drive has a 7-millisecond (ms) seek time and a 3 ms access time. Seek time is the average amount of time it takes the read/write head to move to a location on the platter. Access time is the average amount of time it takes to read or write the data to disk, once the head is in the correct location. Thus, the average time for reading a unique block of data in a 10,000-RPM HD constitutes a seek and an access, for a total of approximately 10 ms (or .010 seconds) per block of data.

  When every disk access requires movement of the head to a new location on the disk, the read/write behavior is referred to as "random." Thus, when all I/O is random, a 10,000-RPM HD can handle approximately 100 I/O per second (IOPS) (the formula is 1000 ms per second divided by 10 ms per I/O or 1000/10=100 IOPS).

  Alternatively, when all I/O occurs from adjacent sectors on the HD, this is referred to as sequential I/O. Sequential I/O has no seek time because when the first I/O is complete, the read/write head is at the start of where the next block of data is stored on the HD. Thus a 10,000-RPM HD is capable of handling approximately 333 I/O per second (1000 ms per second divided by 3 ms per I/O).

  > **NOTE**
  > This example does not reflect the disk cache, where the data of one cylinder is typically kept. In this case, the 10 ms are needed on the first I/O and the disk reads the whole cylinder. All other sequential I/O is satisfied from the cache. As a result, in-disk caches might improve sequential I/O performance.

  So far, the transfer rate of the hard drive has been irrelevant. Whether the hard drive is 20 MB/s Ultra Wide or an Ultra3 160 MB/s, the actual amount of IOPS the can be handled by the 10,000-RPM HD is ~100 random or ~300 sequential I/O. As block sizes change based on the application writing to the drive, the amount of data that is pulled per I/O is different. For example, if the block size is 8 KB, 100 I/O operations will read from or write to the hard drive a total of 800 KB. However, if the block size is 32 KB, 100 I/O will read/write 3,200 KB (3.2 MB) to the hard drive. As long as the SCSI transfer rate is in excess of the total amount of data transferred, getting a "faster" transfer rate drive will gain nothing. See the following tables for comparison.

DESCRIPTION	7200 RPM 9MS SEEK, 4MS ACCESS	10,000 RPM 7MS SEEK, 3MS ACCESS	15,000 RPM 4MS SEEK, 2MS ACCESS
Random I/O	80	100	150
Sequential I/O	250	300	500

10,000 RPM DRIVE	8 KB BLOCK SIZE (ACTIVE DIRECTORY JET)
Random I/O	800 KB/s
Sequential I/O	2400 KB/s

- **SCSI backplane (bus)** – Understanding how the "SCSI backplane (bus)", or in this scenario the ribbon cable, impacts throughput of the storage subsystem depends on knowledge of the block size. Essentially the question would be, how much I/O can the bus handle if the I/O is in 8 KB blocks? In this scenario, the SCSI bus is 20 MB/s, or 20480 KB/s. 20480 KB/s divided by 8 KB blocks yields a maximum of approximately 2500 IOPS supported by the SCSI bus.

> **NOTE**
>
> The figures in the following table represent an example. Most attached storage devices currently use PCI Express, which provides much higher throughput.

I/O SUPPORTED BY SCSI BUS PER BLOCK SIZE	2 KB BLOCK SIZE	8 KB BLOCK SIZE (AD JET) (SQL SERVER 7.0/SQL SERVER 2000)
20 MB/s	10,000	2,500
40 MB/s	20,000	5,000
128 MB/s	65,536	16,384
320 MB/s	160,000	40,000

As can be determined from this chart, in the scenario presented, no matter what the use, the bus will never be a bottleneck, as the spindle maximum is 100 I/O, well below any of the above thresholds.

> **NOTE**
>
> This assumes that the SCSI bus is 100% efficient.

- **SCSI adapter** – For determining the amount of I/O that this can handle, the manufacturer's specifications need to be checked. Directing I/O requests to the appropriate device requires processing of some sort, thus the amount of I/O that can be handled is dependent on the SCSI adapter (or array controller) processor.

  In this example, the assumption that 1,000 I/O can be handled will be made.

- **PCI bus** – This is an often overlooked component. In this example, this will not be the bottleneck; however as systems scale up, it can become a bottleneck. For reference, a 32 bit PCI bus operating at 33Mhz can in theory transfer 133 MB/s of data. Following is the equation:

  ```
  32 bits ÷ 8 bits per byte × 33 MHz = 133 MB/s.
  ```

  Note that is the theoretical limit; in reality only about 50% of the maximum is actually reached, although in certain burst scenarios, 75% efficiency can be obtained for short periods.

  A 66Mhz 64-bit PCI bus can support a theoretical maximum of (64 bits ÷ 8 bits per byte × 66 Mhz) = 528 MB/sec. Additionally, any other device (such as the network adapter, second SCSI controller, and so on) will reduce the bandwidth available as the bandwidth is shared and the devices will contend for the limited resources.

After analysis of the components of this storage subsystem, the spindle is the limiting factor in the amount of I/O that can be requested, and consequently the amount of data that can transit the system. Specifically, in an AD DS scenario, this is 100 random I/O per second in 8 KB increments, for a total of 800 KB per second when accessing the Jet database. Alternatively, the maximum throughput for a spindle that is exclusively allocated to log files would suffer the following limitations: 300 sequential I/O per second in 8 KB increments, for a total of 2400 KB (2.4 MB) per second.

Now, having analyzed a simple configuration, the following table demonstrates where the bottleneck will occur as components in the storage subsystem are changed or added.

NOTES	BOTTLENECK ANALYSIS	DISK	BUS	ADAPTER	PCI BUS
This is the domain controller configuration after adding a second disk. The disk configuration represents the bottleneck at 800 KB/s.	Add 1 disk (Total=2) I/O is random 4 KB block size 10,000 RPM HD	200 I/Os total 800 KB/s total.			
After adding 7 disks, the disk configuration still represents the bottleneck at 3200 KB/s.	**Add 7 disks (Total=8)** I/O is random 4 KB block size 10,000 RPM HD	800 I/Os total. 3200 KB/s total			
After changing I/O to sequential, the network adapter becomes the bottleneck because it is limited to 1000 IOPS.	Add 7 disks (Total=8) **I/O is sequential** 4 KB block size 10,000 RPM HD			2400 I/O sec can be read/written to disk, controller limited to 1000 IOPS	
After replacing the network adapter with a SCSI adapter that supports 10,000 IOPS, the bottleneck returns to the disk configuration.	Add 7 disks (Total=8) I/O is random 4 KB block size 10,000 RPM HD **Upgrade SCSI adapter (now supports 10,000 I/O)**	800 I/Os total. 3,200 KB/s total			

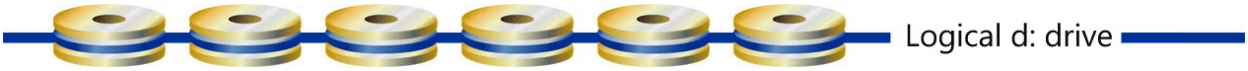
NOTES	BOTTLENECK ANALYSIS	DISK	BUS	ADAPTER	PCI BUS
After increasing the block size to 32 KB, the bus becomes the bottleneck because it only supports 20 MB/s.	Add 7 disks (Total=8) I/O is random **32 KB block size** 10,000 RPM HD		800 I/Os total. 25,600 KB/s (25 MB/s) can be read/written to disk. The bus only supports 20 MB/s		
After upgrading the bus and adding more disks, the disk remains the bottleneck.	**Add 13 disks (Total=14)** Add second SCSI adapter with 14 disks I/O is random 4 KB block size 10,000 RPM HD **Upgrade to 320 MB/s SCSI bus**	2800 I/Os 11,200 KB/s (10.9 MB/s)			
After changing I/O to sequential, the disk remains the bottleneck.	Add 13 disks (Total=14) Add second SCSI Adapter with 14 disks **I/O is sequential** 4 KB block size 10,000 RPM HD Upgrade to 320 MB/s SCSI bus	8,400 I/Os 33,600 KB\s (32.8 MB\s)			

NOTES	BOTTLENECK ANALYSIS	DISK	BUS	ADAPTER	PCI BUS
After adding faster hard drives, the disk remains the bottleneck.	Add 13 disks (Total=14) Add second SCSI adapter with 14 disks I/O is sequential 4 KB block size **15,000 RPM HD** Upgrade to 320 MB/s SCSI bus	14,000 I/Os 56,000 KB/s (54.7 MB/s)			
After increasing the block size to 32 KB, the PCI bus becomes the bottleneck.	Add 13 disks (Total=14) Add second SCSI adapter with 14 disks I/O is sequential **32 KB block size** 15,000 RPM HD Upgrade to 320 MB/s SCSI bus				14,000 I/Os 448,000 KB/s (437 MB/s) is the read/write limit to the spindle. The PCI bus supports a theoretical maximum of 133 MB/s (75% efficient at best).

Introducing RAID

The nature of a storage subsystem does not change dramatically when an array controller is introduced; it just replaces the SCSI adapter in the calculations. What does change is the cost of reading and writing data to the disk when using the various array levels (such as RAID 0, RAID 1, or RAID 5).

In RAID 0, the data is striped across all the disks in the RAID set. This means that during a read or a write operation, a portion of the data is pulled from or pushed to each disk, increasing the amount of data that can transit the system during the same time period. Thus, in one second, on each spindle (again assuming 10,000-RPM drives), 100 I/O operations can be performed. The total amount of I/O that can be supported is N spindles times 100 I/O per second per spindle (yields 100*N I/O per second).



In RAID 1, the data is mirrored (duplicated) across a pair of spindles for redundancy. Thus, when a read I/O operation is performed, data can be read from both of the spindles in the set. This effectively makes the I/O capacity from both disks available during a read operation. The caveat is that write operations gain no performance advantage in a RAID 1. This is because the same data needs to be written to both drives for the sake of redundancy. Though it does not take any longer, as the write of data occurs concurrently on both spindles, because both spindles are occupied duplicating the data, a write I/O operation in essence prevents two

read operations from occurring. Thus, every write I/O costs two read I/O. A formula can be created from that information to determine the total number of I/O operations that are occurring:

> *Read I/O* + 2 × *Write I/O* = *Total available disk I/O consumed*

When the ratio of reads to writes and the number of spindles are known, the following equation can be derived from the above equation to identify the maximum I/O that can be supported by the array:

> *Maximum IOPS per spindle* × 2 spindles × [(*%Reads* + *%Writes*) ÷ (*%Reads* + 2 × *%Writes*)] = *Total IOPS*

RAID 1+ 0, behaves exactly the same as RAID 1 regarding the expense of reading and writing. However, the I/O is now striped across each mirrored set. If

> *Maximum IOPS per spindle* × 2 spindles × [(*%Reads* + *%Writes*) ÷ (*%Reads* + 2 × *%Writes*)] = *Total I/O*

in a RAID 1 set, when a multiplicity (*N*) of RAID 1 sets are striped, the Total I/O that can be processed becomes N × I/O per RAID 1 set:

> *N* × {*Maximum IOPS per spindle* × 2 spindles × [(*%Reads* + *%Writes*) ÷ (*%Reads* + 2 × *%Writes*)] } = *Total IOPS*

In RAID 5, sometimes referred to as *N* + 1 RAID, the data is striped across *N* spindles and parity information is written to the "+ 1" spindle. However, RAID 5 is much more expensive when performing a write I/O than RAID 1 or 1 + 0. RAID 5 performs the following process every time a write I/O is submitted to the array:

1. Read the old data
2. Read the old parity
3. Write the new data
4. Write the new parity

As every write I/O request that is submitted to the array controller by the operating system requires four I/O operations to complete, write requests submitted take four times as long to complete as a single read I/O. To derive a formula to translate I/O requests from the operating system perspective to that experienced by the spindles:

> *Read I/O* + 4 × *Write I/O* = *Total I/O*

Similarly in a RAID 1 set, when the ratio of reads to writes and the number of spindles are known, the following equation can be derived from the above equation to identify the maximum I/O that can be supported by the array (Note that total number of spindles does not include the "drive" lost to parity):

> *IOPS per spindle* × (*Spindles* – 1) × [(*%Reads* + *%Writes*) ÷ (*%Reads* + 4 × *%Writes*)] = *Total IOPS*

Introducing SANs

Expanding the complexity of the storage subsystem, when a SAN is introduced into the environment, the basic principles outlined do not change, however I/O behavior for all of the systems connected to the SAN needs to be taken into account. As one of the major advantages in using a SAN is an additional amount of redundancy over internally or externally attached storage, capacity planning now needs to take into account fault tolerance needs. Also, more components are introduced that need to be evaluated. Breaking a SAN down into the component parts:

- SCSI or Fibre Channel hard drive

- Storage unit channel backplane
- Storage units
- Storage controller module
- SAN switch(es)
- HBA(s)
- The PCI bus

When designing any system for redundancy, additional components are included to accommodate the potential of failure. It is very important, when capacity planning, to exclude the redundant component from available resources. For example, if the SAN has two controller modules, the I/O capacity of one controller module is all that should be used for total I/O throughput available to the system. This is due to the fact that if one controller fails, the entire I/O load demanded by all connected systems will need to be processed by the remaining controller. As all capacity planning is done for peak usage periods, redundant components should not be factored into the available resources and planned peak utilization should not exceed 80% saturation of the system (in order to accommodate bursts or anomalous system behavior). Similarly, the redundant SAN switch, storage unit, and spindles should not be factored into the I/O calculations.

When analyzing the behavior of the SCSI or Fibre Channel hard drive, the method of analyzing the behavior as outlined previously does not change. Although there are certain advantages and disadvantages to each protocol, the limiting factor on a per disk basis is the mechanical limitation of the hard drive.

Analyzing the channel on the storage unit is exactly the same as calculating the resources available on the SCSI bus, or bandwidth (such as 20 MB/s) divided by block size (such as 8 KB). Where this deviates from the simple previous example is in the aggregation of multiple channels. For example, if there are 6 channels, each supporting 20 MB/s maximum transfer rate, the total amount of I/O and data transfer that is available is 100 MB/s (this is correct, it is not 120 MB/s). Again, fault tolerance is a major player in this calculation, in the event of the loss of an entire channel, the system is only left with 5 functioning channels. Thus, to ensure continuing to meet performance expectations in the event of failure, total throughput for all of the storage channels should not exceed 100 MB/s (this assumes load and fault tolerance is evenly distributed across all channels). Turning this into an I/O profile is dependent on the behavior of the application. In the case of Active Directory Jet I/O, this would correlate to approximately 12,500 I/O per second (100 MB/s ÷ 8 KB per I/O).

Next, obtaining the manufacturer's specifications for the controller modules is required in order to gain an understanding of the throughput each module can support. In this example, the SAN has two controller modules that support 7,500 I/O each. The total throughput of the system may be 15,000 IOPS if redundancy is not desired. In calculating maximum throughput in the case of failure, the limitation is the throughput of one controller, or 7,500 IOPS. This threshold is well below the 12,500 IOPS (assuming 4 KB block size) maximum that can be supported by all of the storage channels, and thus, is currently the bottleneck in the analysis. Still for planning purposes, the desired maximum I/O to be planned for would be 10,400 I/O.

When the data exits the controller module, it transits a Fibre Channel connection rated at 1 GB/s (or 1 Gigabit per second). To correlate this with the other metrics, 1 GB/s turns into 128 MB/s (1 GB/s ÷ 8 bits/byte). As this is in excess of the total bandwidth across all channels in the storage unit (100 MB/s), this will not bottleneck the system. Additionally, as this is only one of the two channels (the additional 1 GB/s Fibre Channel connection being for redundancy), if one connection fails, the remaining connection still has enough capacity to handle all the data transfer demanded.

En route to the server, the data will most likely transit a SAN switch. As the SAN switch has to process the incoming I/O request and forward it out the appropriate port, the switch will have a limit to the amount of I/O that can be handled, however, manufacturers specifications will be required to determine what that limit is. For example, if there are two switches and each switch can handle 10,000 IOPS, the total throughput will be 20,000 IOPS. Again, fault tolerance being a concern, if one switch fails, the total throughput of the system will be 10,000 IOPS. As it is desired not to exceed 80% utilization in normal operation, using no more than 8000 I/O should be the target.

Finally, the HBA installed in the server would also have a limit to the amount of I/O that it can handle. Usually, a second HBA is installed for redundancy, but just like with the SAN switch, when calculating maximum I/O that can be handled, the total throughput of $N-1$ HBAs is what the maximum scalability of the system is.

Caching considerations

Caches are one of the components that can significantly impact the overall performance at any point in the storage system. Detailed analysis about caching algorithms is beyond the scope of this article; however, some basic statements about caching on disk subsystems are worth illuminating:

- Caching does improved sustained sequential write I/O as it can buffer many smaller write operations into larger I/O blocks and de-stage to storage in fewer, but larger block sizes. This will reduce total random I/O and total sequential I/O, thus providing more resource availability for other I/O.
- Caching does not improve sustained write I/O throughput of the storage subsystem. It only allows for the writes to be buffered until the spindles are available to commit the data. When all the available I/O of the spindles in the storage subsystem is saturated for long periods, the cache will eventually fill up. In order to empty the cache, enough time between bursts, or extra spindles, need to be allotted in order to provide enough I/O to allow the cache to flush.

  Larger caches only allow for more data to be buffered. This means longer periods of saturation can be accommodated.

  In a normally operating storage subsystem, the operating system will experience improved write performance as the data only needs to be written to cache. Once the underlying media is saturated with I/O, the cache will fill and write performance will return to disk speed.
- When caching read I/O, the scenario where the cache is most advantageous is when the data is stored sequentially on the disk, and the cache can read-ahead (it makes the assumption that the next sector contains the data that will be requested next).
- When read I/O is random, caching at the drive controller is unlikely to provide any enhancement to the amount of data that can be read from the disk. Any enhancement is non-existent if the operating system or application-based cache size is greater than the hardware-based cache size.

  In the case of Active Directory, the cache is only limited by the amount of RAM.

SSD considerations

SSDs are a completely different animal than spindle-based hard disks. Yet the two key criteria remain: "How many IOPS can it handle?" and "What is the latency for those IOPS?" In comparison to spindle-based hard disks, SSDs can handle higher volumes of I/O and can have lower latencies. In general and as of this writing, while SSDs are still expensive in a cost-per-Gigabyte comparison, they are very cheap in terms of cost-per-I/O and deserve significant consideration in terms of storage performance.

Considerations:

- Both IOPS and latencies are very subjective to the manufacturer designs and in some cases have been observed to be poorer performing than spindle based technologies. In short, it is more important to review and validate the manufacturer specs drive by drive and not assume any generalities.
- IOPS types can have very different numbers depending on whether it is read or write. AD DS services, in general, being predominantly read-based, will be less affected than some other application scenarios.
- "Write endurance" – this is the concept that SSD cells will eventually wear out. Various manufacturers deal with this challenge different fashions. At least for the database drive, the predominantly read I/O profile allows for downplaying the significance of this concern as the data is not highly volatile.

Summary

One way to think about storage is picturing household plumbing. Imagine the IOPS of the media that the data is

stored on is the household main drain. When this is clogged (such as roots in the pipe) or limited (it is collapsed or too small), all the sinks in the household back up when too much water is being used (too many guests). This is perfectly analogous to a shared environment where one or more systems are leveraging shared storage on an SAN/NAS/iSCSI with the same underlying media. Different approaches can be taken to resolve the different scenarios:

- A collapsed or undersized drain requires a full scale replacement and fix. This would be similar to adding in new hardware or redistributing the systems using the shared storage throughout the infrastructure.
- A "clogged" pipe usually means identification of one or more offending problems and removal of those problems. In a storage scenario this could be storage or system level backups, synchronized antivirus scans across all servers, and synchronized defragmentation software running during peak periods.

In any plumbing design, multiple drains feed into the main drain. If anything stops up one of those drains or a junction point, only the things behind that junction point back up. In a storage scenario, this could be an overloaded switch (SAN/NAS/iSCSI scenario), driver compatibility issues (wrong driver/HBA Firmware/storport.sys combination), or backup/antivirus/defragmentation. To determine if the storage "pipe" is big enough, IOPS and I/O size needs to be measured. At each joint add them together to ensure adequate "pipe diameter."

Appendix D - Discussion on storage troubleshooting - Environments where providing at least as much RAM as the database size is not a viable option

It is helpful to understand why these recommendations exist so that the changes in storage technology can be accommodated. These recommendations exist for two reasons. The first is isolation of IO, such that performance issues (that is, paging) on the operating system spindle do not impact performance of the database and I/O profiles. The second is that log files for AD DS (and most databases) are sequential in nature, and spindle-based hard drives and caches have a huge performance benefit when used with sequential I/O as compared to the more random I/O patterns of the operating system and almost purely random I/O patterns of the AD DS database drive. By isolating the sequential I/O to a separate physical drive, throughput can be increased. The challenge presented by today's storage options is that the fundamental assumptions behind these recommendations are no longer true. In many virtualized storage scenarios, such as iSCSI, SAN, NAS, and Virtual Disk image files, the underlying storage media is shared across multiple hosts, thus completely negating both the "isolation of IO" and the "sequential I/O optimization" aspects. In fact these scenarios add an additional layer of complexity in that other hosts accessing the shared media can degrade responsiveness to the domain controller.

In planning storage performance, there are three categories to consider: cold cache state, warmed cache state, and backup/restore. The cold cache state occurs in scenarios such as when the domain controller is initially rebooted or the Active Directory service is restarted and there is no Active Directory data in RAM. Warm cache state is where the domain controller is in a steady state and the database is cached. These are important to note as they will drive very different performance profiles, and having enough RAM to cache the entire database does not help performance when the cache is cold. One can consider performance design for these two scenarios with the following analogy, warming the cold cache is a "sprint" and running a server with a warm cache is a "marathon."

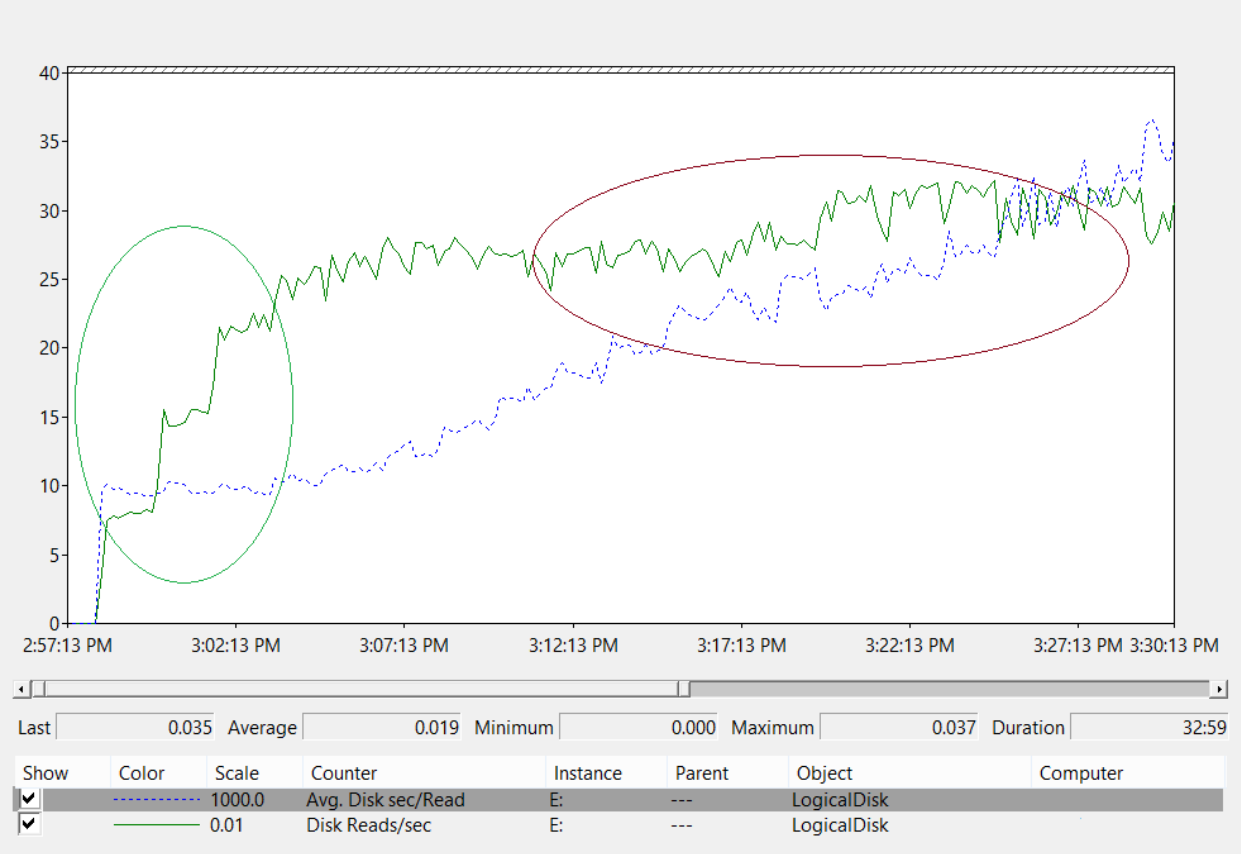
For both the cold cache and warm cache scenario, the question becomes how fast the storage can move the data from disk into memory. Warming the cache is a scenario where, over time, performance improves as more queries reuse data, the cache hit rate increases, and the frequency of needing to go to disk decreases. As a result the adverse performance impact of going to disk decreases. Any degradation in performance is only transient while waiting for the cache to warm and grow to the maximum, system-dependent allowed size. The conversation can be simplified to how quickly the data can be gotten off of disk, and is a simple measure of the IOPS available to Active Directory, which is subjective to IOPS available from the underlying storage. From a

planning perspective, because warming the cache and backup/restore scenarios happen on an exceptional basis, normally occur off hours, and are subjective to the load of the DC, general recommendations do not exist except in that these activities be scheduled for non-peak hours.

AD DS, in most scenarios, is predominantly read IO, usually a ratio of 90% read/10% write. Read I/O often tends to be the bottleneck for user experience, and with write IO, causes write performance to degrade. As I/O to the NTDS.dit is predominantly random, caches tend to provide minimal benefit to read IO, making it that much more important to configure the storage for read I/O profile correctly.

For normal operating conditions, the storage planning goal is minimize the wait times for a request from AD DS to be returned from disk. This essentially means that the number of outstanding and pending I/O is less than or equivalent to the number of pathways to the disk. There are a variety of ways to measure this. In a performance monitoring scenario, the general recommendation is that LogicalDisk(*<NTDS Database Drive>*)\Avg Disk sec/Read be less than 20 ms. The desired operating threshold must be much lower, preferably as close to the speed of the storage as possible, in the 2 to 6 millisecond (.002 to .006 second) range depending on the type of storage.

Example:



Analyzing the chart:

- **Green oval on the left** – The latency remains consistent at 10 ms. The load increases from 800 IOPS to 2400 IOPS. This is the absolute floor to how quickly an I/O request can be processed by the underlying storage. This is subject to the specifics of the storage solution.
- **Burgundy oval on the right** – The throughput remains flat from the exit of the green circle through to the end of the data collection while the latency continues to increase. This is demonstrating that when the request volumes exceed the physical limitations of the underlying storage, the longer the requests spend sitting in the queue waiting to be sent out to the storage subsystem.

Applying this knowledge:

- **Impact to a user querying membership of a large group** – Assume this requires reading 1 MB of data from the disk, the amount of I/O and how long it takes can be evaluated as follows:

- Active Directory database pages are 8 KB in size.
- A minimum of 128 pages need to be read in from disk.
- Assuming nothing is cached, at the floor (10 ms) this is going to take a minimum 1.28 seconds to load the data from disk in order to return it to the client. At 20 ms, where the throughput on storage has long since maxed out and is also the recommended maximum, it will take 2.5 seconds to get the data from disk in order to return it to the end user.

- **At what rate will the cache be warmed** – Making the assumption that the client load is going to maximize the throughput on this storage example, the cache will warm at a rate of 2400 IOPS × 8 KB per IO. Or, approximately 20 MB/s per second, loading about 1 GB of database into RAM every 53 seconds.

> **NOTE**
>
> It is normal for short periods to observe the latencies climb when components aggressively read or write to disk, such as when the system is being backed up or when AD DS is running garbage collection. Additional head room on top of the calculations should be provided to accommodate these periodic events. The goal being to provide enough throughput to accommodate these scenarios without impacting normal function.

As can be seen, there is a physical limit based on the storage design to how quickly the cache can possibly warm. What will warm the cache are incoming client requests up to the rate that the underlying storage can provide. Running scripts to "pre-warm" the cache during peak hours will provide competition to load driven by real client requests. That can adversely affect delivering data that clients need first because, by design, it will generate competition for scarce disk resources as artificial attempts to warm the cache will load data that is not relevant to the clients contacting the DC.

Proper placement of domain controllers and site considerations

12/16/2022 • 6 minutes to read • Edit Online

Proper site definition is critical to performance. Clients falling out of site can experience poor performance for authentications and queries. Furthermore, with the introduction of IPv6 on clients, the request can come from either the IPv4 or the IPv6 address and Active Directory needs to have sites properly defined for IPv6. The operating system prefers IPv6 to IPv4 when both are configured.

Starting in Windows Server 2008, the domain controller attempts to use name resolution to do a reverse lookup in order to determine the site the client should be in. This can cause exhaustion of the ATQ Thread Pool and cause the domain controller to become unresponsive. The appropriate resolution to this is to properly define the site topology for IPv6. As a workaround, one can optimize the name resolution infrastructure to respond quickly to domain controller requests. For more info see Windows Server 2008 or Windows Server 2008 R2 Domain Controller delayed response to LDAP or Kerberos requests.

An additional area of consideration is locating Read/Write DCs for scenarios where RODCs are in use. Certain operations require access to a writable Domain Controller or target a writable Domain Controller when a Read-Only Domain Controller would suffice. Optimizing these scenarios would take two paths:

- Contacting writable Domain Controllers when a Read-Only Domain Controller would suffice. This requires an application code change.
- Where a writable Domain Controller may be necessary. Place read-write Domain Controllers at central locations to minimize latency.

For further information reference:

- Application Compatibility with RODCs
- Active Directory Service Interface (ADSI) and the Read Only Domain Controller (RODC) – Avoiding performance issues

Optimize for referrals

Referrals are how LDAP queries are redirected when the domain controller does not host a copy of the partition queried. When a referral is returned, it contains the distinguished name of the partition, a DNS name, and a port number. The client uses this information to continue the query on a server that hosts the partition. This is a DCLocator scenario and all of the recommendations site definitions and domain controller placement is maintained, but applications which depend on referrals are often overlooked. It is recommended to ensure AD Topology including site definitions and domain controller placement properly reflects the needs of the client. Also, this may include having domain controllers from multiple domains in a single site, tuning DNS settings, or relocating the site of an application.

Optimization considerations for trusts

In an intra-forest scenario, trusts are processed according to the following domain hierarchy: Grand-Child Domain -> Child Domain -> Forest Root Domain -> Child Domain -> Grand-Child Domain. This means that secure channels at the forest root, and each parent, can become overloaded due to aggregation of authentication requests transiting the DCs in the trust hierarchy. This may also incur delays in Active Directories of large geographical dispersion when authentication also has to transit highly latent links to affect the above flow. Overloads can occur in inter-forest and down-level trust scenarios. The following recommendations apply

to all scenarios:

- Properly tune the MaxConcurrentAPI to support the load across the secure channel. For more info, see How to do performance tuning for NTLM authentication by using the MaxConcurrentApi setting.
- Create shortcut trusts as appropriate based on load.
- Ensure that every domain controller in the domain is able to perform name resolution and communicate with the domain controllers in the trusted domain.
- Ensure locality considerations are taken into account for trusts.
- Enable Kerberos where possible and minimize use of the secure channel to reduce risk of running into MaxConcurrentAPI bottlenecks.

Cross domain trust scenarios are an area that has been consistently a pain point for many customers. Name resolution and connectivity issues, often due to firewalls, cause resource exhaustion on the trusting domain controller and impact all clients. Furthermore, an often overlooked scenario is optimizing access to trusted domain controllers. The key areas to ensure this works properly are as follows:

- Ensure the DNS and WINS name resolution that the trusting domain controllers are using can resolve an accurate list of domain controllers for the trusted domain.
    - Statically added records have a tendency to become stale and reintroduce connectivity problems over time. DNS forwards, Dynamic DNS, and merging WINS/DNS infrastructures are more maintainable in the long run.
    - Ensure proper configuration of forwarders, conditional forwards, and secondary copies for both forward and reverse lookup zones for every resource in the environment which a client may need to access. Again, this requires manual maintenance and has a tendency to become stale. Consolidation of infrastructures is ideal.
- Domain controllers in the trusting domain will attempt to locate domain controllers in the trusted domain that are in the same site first and then failback to the generic locators.
    - For more info on how DCLocator works, see Finding a Domain Controller in the Closest Site.
    - Converge site names between the trusted and trusting domains to reflect domain controller in the same location. Ensure subnet and IP address mappings are properly linked to sites in both forests. For more info, see Domain Locator Across a Forest Trust.
    - Ensure ports are open, according to DCLocator needs, for domain controller location. If firewalls exist between the domains, ensure that the firewalls are properly configured for ALL trusts. If firewalls are not open, the trusting domain controller will still attempt to access the trusted domain. If communication fails for any reason, the trusting domain controller will eventually time out the request to the trusted domain controller. However, these time outs can take several seconds per request and can exhaust network ports on the trusting domain controller if the volume of incoming requests is high. The client may experience the waits to timeout at the domain controller as hung threads, which could translate to hung applications (if the application runs the request in the foreground thread). For more info, see How to configure a firewall for domains and trusts.
    - Use DnsAvoidRegisterRecords to eliminate poorly performing or high-latency domain controllers, such as those in satellite sites, from advertising to the generic locators. For more info, see How to optimize the location of a domain controller or global catalog that resides outside of a client's site.

> **NOTE**
> There is a practical limit of about 50 to the number of domain controllers the client can consume. These should be the most site-optimal and highest capacity domain controllers.

    - Consider placing domain controllers from trusted and trusting domains in the same physical location.

For all trust scenarios, credentials are routed according to the domain specified in the authentication requests. This is also true for queries to the LookupAccountName and LsaLookupNames (as well as others, these are just the most commonly used) APIs. When the domain parameters for these APIs are passed a NULL value, the domain controller will attempt to find the account name specified in every trusted domain available.

- Disable checking all available trusts when NULL domain is specified. How to restrict the lookup of isolated names in external trusted domains by using the LsaLookupRestrictIsolatedNameLevel registry entry
- Disable passing authentication requests with NULL domain specified across all available trusts. The Lsass.exe process may stop responding if you have many external trusts on an Active Directory domain controller

Additional References

- Performance tuning Active Directory Servers
- Hardware considerations
- LDAP considerations
- Troubleshooting ADDS performance
- Capacity Planning for Active Directory Domain Services

Hardware considerations in ADDS performance tuning

12/16/2022 • 5 minutes to read • Edit Online

IMPORTANT

The following is a summary of the key recommendations and considerations to optimize server hardware for Active Directory workloads covered in greater depth in the Capacity Planning for Active Directory Domain Services article. Readers are highly encouraged to review Capacity Planning for Active Directory Domain Services for a greater technical understanding and implications of these recommendations.

Avoid going to disk

Active Directory caches as much of the database as memory allows. Fetching pages from memory are orders of magnitude faster than going to physical media, whether the media is spindle or SSD based. Add more memory to minimize disk I/O.

- Active Directory Best Practices recommend putting enough RAM to load the entire DIT into memory, plus accommodate the operating system and other installed applications, such as anti-virus, backup software, monitoring, and so on.
    - For limitations of the legacy platforms, see Memory usage by the Lsass.exe process on domain controllers that are running Windows Server 2003 or Windows 2000 Server.
    - Use the Memory\Long-Term Average Standby Cache Lifetime (s) > 30 minutes performance counter.
- Put the operating system, logs, and the database on separate volumes. If all or the majority of the DIT can be cached, once the cache is warmed and under a steady state, this becomes less relevant and offers a little more flexibility in storage layout. In scenarios where the entire DIT cannot be cached, the importance of splitting the operating system, logs, and database on separate volumes becomes more important.
- Normally, I/O ratios to the DIT are about 90% read and 10% write. Scenarios where write I/O volumes significantly exceed 10% - 20% are considered write-heavy. Write-heavy scenarios do not greatly benefit from the Active Directory cache. To guarantee the transactional durability of data that is written to the directory, Active Directory does not perform disk write caching. Instead, it commits all write operations to the disk before it returns a successful completion status for an operation, unless there is an explicit request not to do this. Therefore, fast disk I/O is important to the performance of write operations to Active Directory. The following are hardware recommendations that might improve performance for these scenarios:
    - Hardware RAID controllers
    - Increase the number of low-latency/high-RPM disks hosting the DIT and log files
    - Write caching on the controller
- Review the disk subsystem performance individually for each volume. Most Active Directory scenarios are predominantly read-based, thus the statistics on the volume hosting the DIT are the most important to inspect. However, do not overlook monitoring the rest of the drives, including the operating system, and log files drives. To determine if the domain controller is properly configured to avoid storage being

the bottleneck for performance, reference the section on Storage Subsystems for standards storage recommendations. Across many environments, the philosophy is to ensure that there is enough head room to accommodate surges or spikes in load. These thresholds are warning thresholds where the head room to accommodate surges or spikes in load becomes constrained and client responsiveness degrades. In short, exceeding these thresholds is not bad in the short term (5 to 15 minutes a few times a day), however a system running sustained with these sorts of statistics is not fully caching the database and may be over taxed and should be investigated.

  - Database ==> Instances(lsass/NTDSA)\I/O Database Reads Averaged Latency < 15ms
  - Database ==> Instances(lsass/NTDSA)\I/O Database Reads/sec < 10
  - Database ==> Instances(lsass/NTDSA)\I/O Log Writes Averaged Latency < 10ms
  - Database ==> Instances(lsass/NTDSA)\I/O Log Writes/sec – informational only.

    To maintain consistency of data, all changes must be written to the log. There is no good or bad number here, it is only a measure of how much the storage is supporting.

- Plan non-core disk I/O loads, such as backup and anti-virus scans, for non-peak load periods. Also, use backup and anti-virus solutions that support the low-priority I/O feature introduced in Windows Server 2008 to reduce competition with I/O needs of Active Directory.

Don't over tax the processors

Processors that don't have enough free cycles can cause long wait times on getting threads on to the processor for execution. Across many environments, the philosophy is to ensure that there is enough head room to accommodate surges or spikes in load to minimize impact on client responsiveness in these scenarios. In short, exceeding the below thresholds is not bad in the short term (5 to 15 minutes a few times a day), however a system running sustained with these sorts of statistics doesn't provide any head room to accommodate abnormal loads and can easily be put into an over taxed scenario. Systems spending sustained periods above the thresholds should be investigated to how to reduce processor loads.

- For more info on how to select a processor, see Performance Tuning for Server Hardware.
- Add hardware, optimize load, direct clients elsewhere, or remove load from the environment to reduce CPU load.
- Use the Processor Information(_Total)\% Processor Utilization < 60% performance counter.

Avoid overloading the network adapter

Just like with processors, excessive network adapter utilization will cause long wait times for the outbound traffic to get on to the network. Active Directory tends to have small inbound requests and relatively to significantly larger amounts of data returned to the client systems. Sent data far exceeds received data. Across many environments, the philosophy is to ensure that there is enough head room to accommodate surges or spikes in load. This threshold is a warning threshold where the head room to accommodate surges or spikes in load becomes constrained and client responsiveness degrades. In short, exceeding these thresholds is not bad in the short term (5 to 15 minutes a few times a day), however a system running sustained with these sorts of statistics is over taxed and should be investigated.

- For more info on how to tune the network subsystem, see Performance Tuning for Network Subsystems.
- Use the Compare NetworkInterface(*)\Bytes Sent/Sec with NetworkInterface(*)\Current Bandwidth performance counter. The ratio should be less than 60% utilized.

Additional References

- Performance tuning Active Directory Servers
- LDAP considerations
- Proper placement of domain controllers and site considerations
- Troubleshooting ADDS performance
- Capacity Planning for Active Directory Domain Services

Memory usage considerations for AD DS performance tuning

12/16/2022 • 5 minutes to read • Edit Online

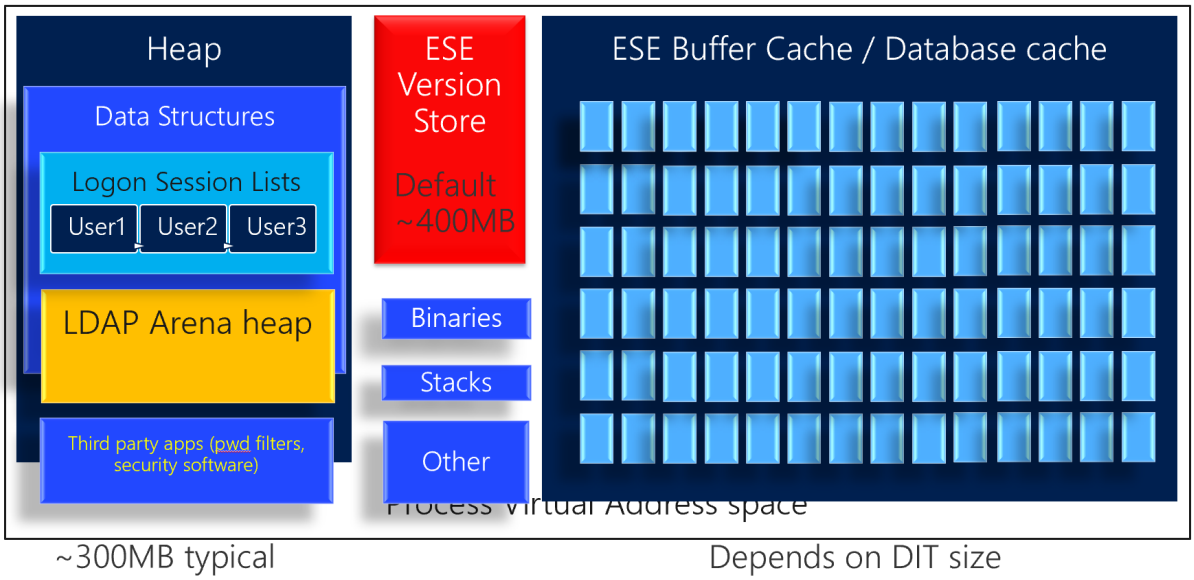
This article describes some basics of the Local Security Authority Subsystem Service (LSASS, also known as the Lsass.exe process), best practices for the configuration of LSASS, and expectations for memory usage. This article should be used as a guide in the analysis of LSASS performance and memory use on domain controllers (DCs). The information in this article may be useful if you have questions about how to tune and configure servers and DCs to optimize this engine.

LSASS is responsible for management of local security authority (LSA) domain authentication and Active Directory management. LSASS handles authentication for both the client and the server, and it also governs the Active Directory engine. LSASS is responsible for the following components:

- Local Security Authority
- NetLogon service
- Security Accounts Manager (SAM) service
- LSA Server service
- Secure Sockets Layer (SSL)
- Kerberos v5 authentication protocol
- NTLM authentication protocol
- Other authentication packages that load into LSA

The Active Directory database services (NTDSAI.dll) work with the Extensible Storage Engine (ESE, ESENT.dll).

Here is a visual diagram of LSASS memory usage on a DC:



The amount of memory that LSASS uses on a DC increases in accordance with Active Directory usage. When data is queried, it is cached in memory. As a result, it is normal to see LSASS using an amount of memory that is larger than the size of the Active Directory database file (NTDS.dit).

As illustrated in the diagram, LSASS memory usage can be divided into several parts, including the ESE database buffer cache, the ESE version store, and others. The rest of this article provides insight into each of these parts.

ESE database buffer cache

The largest variable memory usage within LSASS is the ESE database buffer cache. The size of the cache can range from less than 1 MB to the size of the entire database. Because a larger cache improves performance, the database engine for Active Directory (ESENT) attempts to keep the cache as large as possible. While the size of the cache varies with memory pressure in the computer, the maximum size of the ESE database buffer cache is *only* limited by physical RAM installed in the computer. As long as there is no other memory pressure, the cache can grow to the size of the Active Directory NTDS.dit database file. The more of the database that can be cached, the better the performance of the DC will be.

> **NOTE**
>
> Because of the way that the database caching algorithm works, on a 64-bit system on which the database size is smaller than the available RAM, the database cache can grow larger than the database size by 30 to 40 percent.

ESE version store

There is variable memory usage by the ESE version store (the red part in the diagram above). The amount of memory used depends on whether you have Windows Server 2019 or older versions of Windows.

- In Windows Server versions predating Windows Server 2019, by default LSASS may use up to around 400MB of memory (depending on the number of CPUs) on a 64-bit machine for the ESE version store. For more information about how the version store is used see the following ASKDS blog post by Ryan Ries: The Version Store Called and They're All Out of Buckets.
- In Windows Server 2019, this is simplified and when NTDS service first starts, the ESE version store size is now calculated as 10% of physical RAM, with a minimum of 400MB and a maximum of 4GB. For great details about this and version store troubleshooting, see another great blog from Ryan Ries: Deep Dive: Active Directory ESE Version Store Changes in Server 2019.

Other memory use

Finally, there is code, stacks, heaps, and various fixed size data structures (for example, the schema cache). The amount of memory that LSASS uses may vary, depending on the load on the computer. As the number of running threads increases, so does the number of memory stacks. On average, LSASS uses 100 MB to 300 MB of memory for these fixed components. When a larger amount of RAM is installed, LSASS can use more RAM and less virtual memory.

Limit or minimize the number of programs on your domain controller OR add additional RAM where appropriate

For optimum performance, LSASS takes as much RAM as possible on a given DC. LSASS relinquishes that RAM as other processes ask for it. The idea is to optimize performance of LSASS while still accounting for other processes that might run on a computer. The list of programs to watch out for includes monitoring agents. Some customers have separate agents for various server functions which can consume considerable RAM resources. Some may issue many WMI queries, for which we have a few details below.

Because of this and to increase performance, it is a good practice to limit or minimize the number of programs on a DC. If there are no memory requests, LSASS uses this memory to cache the Active Directory database and therefore achieve optimum performance.

When you notice that a DC has performance problems, also watch for processes with significant memory utilization. These may have a problem you need to troubleshoot. They may include Microsoft components. Please make sure you keep up with recent servicing updates—Microsoft includes solutions for excessive memory utilization as part of the quality updates, which may also help your DC performance.

There are built-in OS facilities that can consume significant RAM depending on the usage profile:

- **File server**. DCs are also file servers for SYSVOL and Netlogon shares, servicing group policy and scripts for policy and startup/logon scripts. However, we see customers use DCs to service other file content. The SMB file server would then consume RAM to track the active clients, but foremost, the file content would make the OS file cache grow, and compete with the ESE database cache for RAM.
- **WMI queries**. Monitoring solutions often make many WMI queries. An individual query may be cheap to execute. Often it is the volume of calls that incurs some overhead, especially as the monitoring solutions extract new events from the various event logs that Windows manages.

  The event log that produces the most volume is typically the Security Event log. And this is also the event log that security administrators want to collect, especially from DCs.

  The WMI service uses a dynamic memory allocation scheme that optimizes queries. Therefore, the WMI service may allocate a lot of memory, again competing with the ESE database cache.

LDAP considerations in ADDS performance tuning

12/16/2022 • 4 minutes to read • Edit Online

IMPORTANT

The following is a summary of the key recommendations and considerations to optimize server hardware for Active Directory workloads covered in greater depth in the Capacity Planning for Active Directory Domain Services article. Readers are highly encouraged to review Capacity Planning for Active Directory Domain Services for a greater technical understanding and implications of these recommendations.

Verify LDAP queries

Verify that LDAP queries conform with the creating efficient queries recommendations.

There is extensive documentation on MSDN about how to properly write, structure, and analyze queries for use against Active Directory. For more info, see Creating More Efficient Microsoft Active Directory-Enabled Applications.

Optimize LDAP page sizes

When returning results with multiple objects in response to client requests, the domain controller has to temporarily store the result set in memory. Increasing page sizes will cause more memory usage and can age items out of cache unnecessarily. In this case, the default settings are optimal. There are several scenarios where recommendations were made to increase the page size settings. We recommend using the default values unless specifically identified as inadequate.

When queries have many results, a limit of similar queries concurrently executed may be encountered. This occurs as the LDAP server may deplete a global memory area known as the cookie pool. It may be necessary to increase the size of the pool as discussed in How LDAP Server Cookies Are Handled.

To tune these settings, see Windows Server 2008 and newer domain controller returns only 5000 values in a LDAP response.

Determine whether to add indices

Indexing attributes is useful when searching for objects that have the attribute name in a filter. Indexing can reduce the number of objects that must be visited when evaluating the filter. However, this reduces the performance of write operations because the index must be updated when the corresponding attribute is modified or added. It also increases the size of the directory database, though the benefits often outweigh the cost of storage. Logging can be used to find the expensive and inefficient queries. Once identified, consider indexing some attributes that are used in the corresponding queries to improve the search performance. For more info on how Active Directory Searches work, see How Active Directory Searches Work.

Scenarios that benefit in adding indices

- Client load in requesting the data is generating significant CPU usage and the client query behavior cannot be changed or optimized.
- The client load is generating significant disk I/O on a server due to an unindexed attribute and the client query behavior cannot be changed or optimized.
- A query is taking a long time and is not completing in an acceptable timeframe to the client due to lack of

covering indices.

- Large volumes of queries with high durations are causing consumption and exhaustion of ATQ LDAP Threads. Monitor the following performance counters:
  - **NTDS\Request Latency** – This is subject to how long the request takes to process. Active Directory times out requests after 120 seconds (default), however, the majority should run much faster and extremely long running queries should get hidden in the overall numbers. Look for changes in this baseline, rather than absolute thresholds.

    > **NOTE**
    > High values here can also be indicators of delays in "proxying" requests to other domains and CRL checks.

  - **NTDS\Estimated Queue Delay** – This should ideally be near 0 for optimal performance as this means that requests spend no time waiting to be serviced.

These scenarios can be detected using one or more of the following approaches:

- Determining Query Timing with the Statistics Control
- Tracking Expensive and Inefficient Searches
- Active Directory Diagnostics Data Collector Set in Performance Monitor (Son of SPA: AD Data Collector Sets in Win2008 and beyond)
- Searches using any filter besides "(objectClass=*)" that use the Ancestors Index.

Other index considerations

- Ensure that creating the index is the right solution to the problem after tuning the query has been exhausted as an option. Sizing hardware properly is very important. Indices should be added only when the right fix is to index the attribute, and not an attempt to obfuscate hardware problems.
- Indices increase the size of the database by a minimum of the total size of the attribute being indexed. An estimate of database growth can therefore be evaluated by taking the average size of the data in the attribute and multiplying by the number of objects that will have the attribute populated. Generally this is about a 1% increase in database size. For more info, see How the Data Store Works.
- If search behavior is predominantly done at the organization unit level, consider indexing for containerized searches.
- Tuple indices are larger than normal indices, but it is much harder to estimate the size. Use normal indices size estimates as the floor for growth, with a maximum of 20%. For more info, see How the Data Store Works.
- If search behavior is predominantly done at the organization unit level, consider indexing for containerized searches.
- Tuple Indices are needed to support medial search strings and final search strings. Tuple indices are not needed for initial search strings.
  - Initial Search String – (samAccountName=MYPC*)
  - Medial Search String - (samAccountName=*MYPC*)
  - Final Search String – (samAccountName=*MYPC$)
- Creating an index will generate disk I/O while the index is being built. This is done on a background thread with lower priority and incoming requests will be prioritized over the index build. If capacity

planning for the environment has been done correctly, this should be transparent. However, write-heavy scenarios or an environment where the load on the domain controller storage is unknown could degrade client experience and should be done off-hours.

- Affects to replication traffic is minimal since building indices occurs locally.

For more info, see the following:

- Creating More Efficient Microsoft Active Directory-Enabled Applications
- Searching in Active Directory Domain Services
- Indexed Attributes

Additional References

- Performance tuning Active Directory Servers
- Hardware considerations
- Proper placement of domain controllers and site considerations
- Troubleshooting ADDS performance
- Capacity Planning for Active Directory Domain Services

Troubleshooting Active Directory Domain Services performance

12/16/2022 • 2 minutes to read • Edit Online

For additional information on ADDS performance troubleshooting, see Monitoring Your Branch Office Environment.

Additional References

- Performance tuning Active Directory Servers
- Hardware considerations
- LDAP considerations
- Proper placement of domain controllers and site considerations
- Capacity Planning for Active Directory Domain Services

Performance tuning for file servers

12/16/2022 • 7 minutes to read • Edit Online

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.

General tuning parameters for clients

The following REG_DWORD registry settings can affect the performance of client computers that interact with SMB file servers:

- **ConnectionCountPerNetworkInterface**

  ```
  HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\ConnectionCountPerNetworkInterface
  ```

  Applies to Windows 10, Windows 8.1, Windows 8, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, and Windows Server 2012

  The default is 1, and we strongly recommend using the default. The valid range is 1-16. The maximum number of connections per interface to be established with a server for non-RSS interfaces.

- **ConnectionCountPerRssNetworkInterface**

  ```
  HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\ConnectionCountPerRssNetworkInterface
  ```

  Applies to Windows 10, Windows 8.1, Windows 8, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, and Windows Server 2012

  The default is 4, and we strongly recommend using the default. The valid range is 1-16. The maximum number of connections per interface to be established with a server for RSS interfaces.

- **ConnectionCountPerRdmaNetworkInterface**

  ```
  HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\ConnectionCountPerRdmaNetworkInterface
  ```

  Applies to Windows 10, Windows 8.1, Windows 8, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, and Windows Server 2012

  The default is 2, and we strongly recommend using the default. The valid range is 1-16. The maximum number of connections per interface to be established with a server for RDMA interfaces.

- **MaximumConnectionCountPerServer**

  ```
  HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\MaximumConnectionCountPerServer
  ```

  Applies to Windows 10, Windows 8.1, Windows 8, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, and Windows Server 2012

The default is 32, with a valid range from 1-64. The maximum number of connections to be established with a single server running Windows Server 2012 across all interfaces.

- **DormantDirectoryTimeout**

```
HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\DormantDirectoryTimeout
```

Applies to Windows 10, Windows 8.1, Windows 8, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, and Windows Server 2012

The default is 600 seconds. The maximum time server directory handles held open with directory leases.

- **FileInfoCacheLifetime**

```
HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\FileInfoCacheLifetime
```

Applies to Windows 10, Windows 8.1, Windows 8, Windows 7, Windows Vista, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, and Windows Server 2008

The default is 10 seconds. The file information cache timeout period.

- **DirectoryCacheLifetime**

```
HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\DirectoryCacheLifetime
```

Applies to Windows 10, Windows 8.1, Windows 8, Windows 7, Windows Vista, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, and Windows Server 2008

The default is 10 seconds. This is the directory cache timeout.

> **NOTE**
> This parameter controls caching of directory metadata in the absence of directory leases.

> **NOTE**
> A known issue in Windows 10, version 1803, affects the ability of Windows 10 to cache large directories. After you upgrade a computer to Windows 10, version 1803, you access a network share that contains thousands of files and folders, and you open a document that is located on that share. During both of these operations, you experience significant delays.
>
> To resolve this issue, install Windows 10, version 1809 or a later version.
>
> To work around this issue, set **DirectoryCacheLifetime** to **0**.
>
> This issue affects the following editions of Windows 10:
>
> - Windows 10 Enterprise, version 1803
> - Windows 10 Pro for Workstations, version 1803
> - Windows 10 Pro Education, version 1803
> - Windows 10 Professional, version 1803
> - Windows 10 Education, version 1803
> - Windows 10 Home, version 1803

- **DirectoryCacheEntrySizeMax**

  ```
  HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\DirectoryCacheEntrySizeMax
  ```

  Applies to Windows 10, Windows 8.1, Windows 8, Windows 7, Windows Vista, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, and Windows Server 2008

  The default is 64 KB. This is the maximum size of directory cache entries.

- **FileNotFoundCacheLifetime**

  ```
  HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\FileNotFoundCacheLifetime
  ```

  Applies to Windows 10, Windows 8.1, Windows 8, Windows 7, Windows Vista, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, and Windows Server 2008

  The default is 5 seconds. The file not found cache timeout period.

- **CacheFileTimeout**

  ```
  HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\CacheFileTimeout
  ```

  Applies to Windows 8.1, Windows 8, Windows Server 2012, Windows Server 2012 R2, and Windows 7

  The default is 10 seconds. This setting controls the length of time (in seconds) that the redirector will hold on to cached data for a file after the last handle to the file is closed by an application.

- **DisableBandwidthThrottling**

  ```
  HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\DisableBandwidthThrottling
  ```

  Applies to Windows 10, Windows 8.1, Windows 8, Windows 7, Windows Vista, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, and Windows Server 2008

  The default is 0. By default, the SMB redirector throttles throughput across high-latency network connections, in some cases to avoid network-related timeouts. Setting this registry value to 1 disables this throttling, enabling higher file transfer throughput over high-latency network connections.

- **DisableLargeMtu**

  ```
  HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\DisableLargeMtu
  ```

  Applies to Windows 10, Windows 8.1, Windows 8, Windows 7, Windows Vista, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, and Windows Server 2008

  The default is 0 for Windows 8 only. In Windows 8, the SMB redirector transfers payloads as large as 1 MB per request, which can improve file transfer speed. Setting this registry value to 1 limits the request size to 64 KB. You should evaluate the impact of this setting before applying it.

- **RequireSecuritySignature**

```
HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\RequireSecuritySignature
```

Applies to Windows 10, Windows 8.1, Windows 8, Windows 7, Windows Vista, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, and Windows Server 2008

The default is 0, disabling SMB Signing. Changing this value to 1 enables SMB signing for all SMB communication, preventing SMB communication with computers where SMB signing is disabled. SMB signing can increase CPU cost and network round trips, but helps block man-in-the-middle attacks. If SMB signing is not required, ensure that this registry value is 0 on all clients and servers.

For more info, see The Basics of SMB Signing.

- **FileInfoCacheEntriesMax**

```
HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\FileInfoCacheEntriesMax
```

Applies to Windows 10, Windows 8.1, Windows 8, Windows 7, Windows Vista, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, and Windows Server 2008

The default is 64, with a valid range of 1 to 65536. This value is used to determine the amount of file metadata that can be cached by the client. Increasing the value can reduce network traffic and increase performance when a large number of files are accessed.

- **DirectoryCacheEntriesMax**

```
HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\DirectoryCacheEntriesMax
```

Applies to Windows 10, Windows 8.1, Windows 8, Windows 7, Windows Vista, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, and Windows Server 2008

The default is 16, with a valid range of 1 to 4096. This value is used to determine the amount of directory information that can be cached by the client. Increasing the value can reduce network traffic and increase performance when large directories are accessed.

- **FileNotFoundCacheEntriesMax**

```
HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\FileNotFoundCacheEntriesMax
```

Applies to Windows 10, Windows 8.1, Windows 8, Windows 7, Windows Vista, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, and Windows Server 2008

The default is 128, with a valid range of 1 to 65536. This value is used to determine the amount of file name information that can be cached by the client. Increasing the value can reduce network traffic and increase performance when a large number of file names are accessed.

- **MaxCmds**

```
HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\MaxCmds
```

Applies to Windows 10, Windows 8.1, Windows 8, Windows 7, Windows Vista, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, and Windows Server 2008

The default is 15. This parameter limits the number of outstanding requests on a session. Increasing the value can use more memory, but it can improve performance by enabling a deeper request pipeline. Increasing the value in conjunction with MaxMpxCt can also eliminate errors that are encountered due to large numbers of outstanding long-term file requests, such as FindFirstChangeNotification calls. This parameter does not affect connections with SMB 2.0 servers.

- DormantFileLimit

```
HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters\DormantFileLimit
```

Applies to Windows 10, Windows 8.1, Windows 8, Windows 7, Windows Vista, Windows Server 2022, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, and Windows Server 2008

The default is 1023. This parameter specifies the maximum number of files that should be left open on a shared resource after the application has closed the file.

Client tuning example

The general tuning parameters for client computers can optimize a computer for accessing remote file shares, particularly over some high-latency networks (such as branch offices, cross-datacenter communication, home offices, and mobile broadband). The settings are not optimal or appropriate on all computers. You should evaluate the impact of individual settings before applying them.

PARAMETER	VALUE	DEFAULT
DisableBandwidthThrottling	1	0
FileInfoCacheEntriesMax	32768	64
DirectoryCacheEntriesMax	4096	16
FileNotFoundCacheEntriesMax	32768	128
MaxCmds	32768	15

Starting in Windows 8, you can configure many of these SMB settings by using the **Set-SmbClientConfiguration** and **Set-SmbServerConfiguration** Windows PowerShell cmdlets. Registry-only settings can be configured by using Windows PowerShell as well.

```
Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\LanmanWorkstation\Parameters"
RequireSecuritySignature -Value 0 -Force
```

Performance tuning for SMB file servers

12/16/2022 • 8 minutes to read • Edit Online

SMB configuration considerations

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

Ensure that the BIOS and operating system power management modes are set as needed, which might include High Performance mode or altered C-State. Ensure that the latest, most resilient, and fastest storage and networking device drivers are installed.

Copying files is a common operation performed on a file server. Windows Server has several built-in file copy utilities that you can run by using a command prompt. Robocopy is recommended. Introduced in Windows Server 2008 R2, the **/mt** option of Robocopy can significantly improve speed on remote file transfers by using multiple threads when copying multiple small files. We also recommend using the **/log** option to reduce console output by redirecting logs to a NUL device or to a file. When you use Xcopy, we recommend adding the **/q** and **/k** options to your existing parameters. The former option reduces CPU overhead by reducing console output and the latter reduces network traffic.

SMB performance tuning

File server performance and available tunings depend on the SMB protocol that is negotiated between each client and the server, and on the deployed file server features. The highest protocol version currently available is SMB 3.1.1 in Windows Server 2022, Windows Server 2016 and Windows 10. You can check which version of SMB is in use on your network by using Windows PowerShell **Get-SMBConnection** on clients and **Get-SMBSession | FL** on servers.

SMB 3.0 protocol family

SMB 3.0 was introduced in Windows Server 2012 and further enhanced in Windows Server 2012 R2 (SMB 3.02) and Windows Server 2016 (SMB 3.1.1). This version introduced technologies that may significantly improve performance and availability of the file server. For more info, see SMB in Windows Server 2012 and 2012 R2 2012 and What's new in SMB 3.1.1.

SMB Direct

SMB Direct introduced the ability to use RDMA network interfaces for high throughput with low latency and low CPU utilization.

Whenever SMB detects an RDMA-capable network, it automatically tries to use the RDMA capability. However, if for any reason the SMB client fails to connect using the RDMA path, it will simply continue to use TCP/IP connections instead. All RDMA interfaces that are compatible with SMB Direct are required to also implement a TCP/IP stack, and SMB Multichannel is aware of that.

SMB Direct is not required in any SMB configuration, but it' s always recommended for those who want lower latency and lower CPU utilization.

For more info about SMB Direct, see Improve Performance of a File Server with SMB Direct.

SMB Multichannel

SMB Multichannel allows file servers to use multiple network connections simultaneously and provides increased throughput.

For more info about SMB Multichannel, see Deploy SMB Multichannel.

SMB Scale-Out

SMB Scale-out allows SMB 3.0 in a cluster configuration to show a share in all nodes of a cluster. This active/active configuration makes it possible to scale file server clusters further, without a complex configuration with multiple volumes, shares and cluster resources. The maximum share bandwidth is the total bandwidth of all file server cluster nodes. The total bandwidth is no longer limited by the bandwidth of a single cluster node, but rather depends on the capability of the backing storage system. You can increase the total bandwidth by adding nodes.

For more info about SMB Scale-Out, see Scale-Out File Server for Application Data Overview and the blog post To scale out or not to scale out, that is the question.

Performance counters for SMB 3.0

The following SMB performance counters were introduced in Windows Server 2012, and they are considered a base set of counters when you monitor the resource usage of SMB 2 and higher versions. Log the performance counters to a local, raw (.blg) performance counter log. It is less expensive to collect all instances by using the wildcard character (*), and then extract particular instances during post-processing by using Relog.exe.

- **SMB Client Shares**

  These counters display information about file shares on the server that are being accessed by a client that is using SMB 2.0 or higher versions.

  If you' re familiar with the regular disk counters in Windows, you might notice a certain resemblance. That' s not by accident. The SMB client shares performance counters were designed to exactly match the disk counters. This way you can easily reuse any guidance on application disk performance tuning you currently have. For more info about counter mapping, see Per share client performance counters blog.

- **SMB Server Shares**

  These counters display information about the SMB 2.0 or higher file shares on the server.

- **SMB Server Sessions**

  These counters display information about SMB server sessions that are using SMB 2.0 or higher.

  Turning on counters on server side (server shares or server sessions) may have significant performance impact for high IO workloads.

- **Resume Key Filter**

  These counters display information about the Resume Key Filter.

- **SMB Direct Connection**

  These counters measure different aspects of connection activity. A computer can have multiple SMB Direct connections. The SMB Direct Connection counters represent each connection as a pair of IP addresses and ports, where the first IP address and port represent the connection's local endpoint, and the second IP address and port represent the connection's remote endpoint.

- **Physical Disk, SMB, CSV FS performance counters relationships**

  For more info on how Physical Disk, SMB, and CSV FS (file system) counters are related, see the following blog post: Cluster Shared Volume Performance Counters.

Tuning parameters for SMB file servers

The following REG_DWORD registry settings can affect the performance of SMB file servers:

- **Smb2CreditsMin** and **Smb2CreditsMax**

```
HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters\Smb2CreditsMin
```

```
HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters\Smb2CreditsMax
```

The defaults are 512 and 8192, 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, for example, copying files over high-bandwidth, high-latency links.

> **TIP**
> Prior to Windows 10 and Windows Server 2016, the number of credits granted to the client varied dynamically between Smb2CreditsMin and Smb2CreditsMax based on an algorithm that attempted to determine the optimal number of credits to grant based on network latency and credit usage. In Windows 10 and Windows Server 2016, the SMB server was changed to unconditionally grant credits upon request up to the configured maximum number of credits. As part of this change, the credit throttling mechanism, which reduces the size of each connection's credit window when the server is under memory pressure, was removed. The kernel's low memory event that triggered throttling is only signaled when the server is so low on memory (< a few MB) as to be useless. Since the server no longer shrinks credit windows the Smb2CreditsMin setting is no longer necessary and is now ignored.
>
> You can monitor SMB Client Shares\Credit Stalls /Sec to see if there are any issues with credits.

- **AdditionalCriticalWorkerThreads**

```
HKLM\System\CurrentControlSet\Control\Session Manager\Executive\AdditionalCriticalWorkerThreads
```

The default is 0, which means that no additional critical kernel worker threads are added. This value affects the number of threads that the file system cache uses for read-ahead and write-behind requests. Raising this value can allow for more queued I/O in the storage subsystem, and it can improve I/O performance, particularly on systems with many logical processors and powerful storage hardware.

> **TIP**
> The value may need to be increased if the amount of cache manager dirty data (performance counter Cache\Dirty Pages) is growing to consume a large portion (over ~25%) of memory or if the system is doing lots of synchronous read I/Os.

- **MaxThreadsPerQueue**

```
HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters\MaxThreadsPerQueue
```

The default is 20. Increasing this value raises the number of threads that the file server can use to service concurrent requests. When a large number of active connections need to be serviced, and hardware resources, such as storage bandwidth, are sufficient, increasing the value can improve server scalability, performance, and response times.

> **TIP**
> An indication that the value may need to be increased is if the SMB2 work queues are growing very large (performance counter 'Server Work Queues\Queue Length\SMB2 NonBlocking *' is consistently above ~100).

> **NOTE**
> In Windows 10, Windows Server 2016, and Windows Server 2022, MaxThreadsPerQueue is unavailable. The number of threads for a thread pool will be "20 * the number of processors in a NUMA node".

- AsynchronousCredits

```
HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters\AsynchronousCredits
```

The default is 512. This parameter limits the number of concurrent asynchronous SMB commands that are allowed on a single connection. Some cases (such as when there is a front-end server with a back-end IIS server) require a large amount of concurrency (for file change notification requests, in particular). The value of this entry can be increased to support these cases.

- RemoteFileDirtyPageThreshold

```
HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\RemoteFileDirtyPageThreshold
```

The default is 5GB. This value determines the maximum number of dirty pages in the cache (on a per-file basis) for a remote write before an inline flush will be performed. We do not recommend changing this value unless the system experiences consistent slowdowns during heavy remote writes. This slowdown behavior would typically be seen where the client has faster storage IO performance than the remote server. The setting change is applied to the server. Client and server refer to the distributed system architecture, not to particular operating systems; for example, a Windows Server copying data to another Windows Server over SMB would still involve an SMB client and an SMB server. See Troubleshoot Cache and Memory Manager Performance Issues for more information.

SMB server tuning example

The following settings can optimize a computer for file server performance in many cases. The settings are not optimal or appropriate on all computers. You should evaluate the impact of individual settings before applying them.

PARAMETER	VALUE	DEFAULT
AdditionalCriticalWorkerThreads	64	0
MaxThreadsPerQueue	64	20

SMB client performance monitor counters

For more info about SMB client counters, see Windows Server 2012 File Server Tip: New per-share SMB client performance counters provide great insight.

Performance Tuning NFS File Servers

12/16/2022 • 5 minutes to read • Edit Online

Services for NFS model

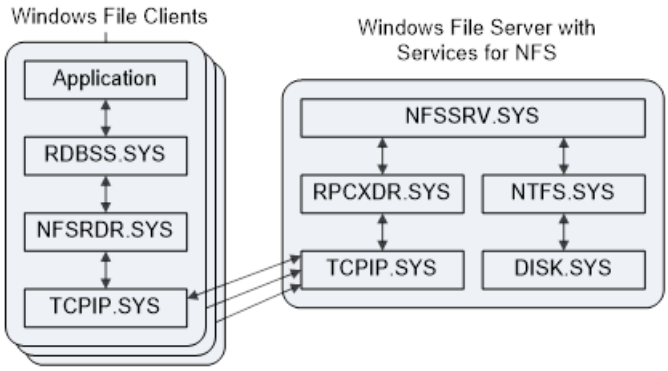
The following sections provide information about the Microsoft Services for Network File System (NFS) model for client-server communication. Since NFS v2 and NFS v3 are still the most widely deployed versions of the protocol, all of the registry keys except for MaxConcurrentConnectionsPerIp apply to NFS v2 and NFS v3 only.

No registry tuning is required for NFS v4.1 protocol.

Service for NFS model overview

Microsoft Services for NFS provides a file-sharing solution for enterprises that have a mixed Windows and UNIX environment. This communication model consists of client computers and a server. Applications on the client request files that are located on the server through the redirector (Rdbss.sys) and NFS mini-redirector (Nfsrdr.sys). The mini-redirector uses the NFS protocol to send its request through TCP/IP. The server receives multiple requests from the clients through TCP/IP and routes the requests to the local file system (Ntfs.sys), which accesses the storage stack.

The following figure shows the communication model for NFS.



Tuning parameters for NFS file servers

The following REG_DWORD registry settings can affect the performance of NFS file servers:

- OptimalReads

  ```
  HKLM\System\CurrentControlSet\Services\NfsServer\Parameters\OptimalReads
  ```

  The default is 0. This parameter determines whether files are opened for FILE_RANDOM_ACCESS or for FILE_SEQUENTIAL_ONLY, depending on the workload I/O characteristics. Set this value to 1 to force files to be opened for FILE_RANDOM_ACCESS. FILE_RANDOM_ACCESS prevents the file system and cache manager from prefetching.

  > **NOTE**
  > This setting must be carefully evaluated because it may have potential impact on system file cache grow.

- RdWrHandleLifeTime

```
HKLM\System\CurrentControlSet\Services\NfsServer\Parameters\RdWrHandleLifeTime
```

The default is 5. This parameter controls the lifetime of an NFS cache entry in the file handle cache. The parameter refers to cache entries that have an associated open NTFS file handle. Actual lifetime is approximately equal to RdWrHandleLifeTime multiplied by RdWrThreadSleepTime. The minimum is 1 and the maximum is 60.

- **RdWrNfsHandleLifeTime**

```
HKLM\System\CurrentControlSet\Services\NfsServer\Parameters\RdWrNfsHandleLifeTime
```

The default is 5. This parameter controls the lifetime of an NFS cache entry in the file handle cache. The parameter refers to cache entries that do not have an associated open NTFS file handle. Services for NFS uses these cache entries to store file attributes for a file without keeping an open handle with the file system. Actual lifetime is approximately equal to RdWrNfsHandleLifeTime multiplied by RdWrThreadSleepTime. The minimum is 1 and the maximum is 60.

- **RdWrNfsReadHandlesLifeTime**

```
HKLM\System\CurrentControlSet\Services\NfsServer\Parameters\RdWrNfsReadHandlesLifeTime
```

The default is 5. This parameter controls the lifetime of an NFS read cache entry in the file handle cache. Actual lifetime is approximately equal to RdWrNfsReadHandlesLifeTime multiplied by RdWrThreadSleepTime. The minimum is 1 and the maximum is 60.

- **RdWrThreadSleepTime**

```
HKLM\System\CurrentControlSet\Services\NfsServer\Parameters\RdWrThreadSleepTime
```

The default is 5. This parameter controls the wait interval before running the cleanup thread on the file handle cache. The value is in ticks, and it is non-deterministic. A tick is equivalent to approximately 100 nanoseconds. The minimum is 1 and the maximum is 60.

- **FileHandleCacheSizeinMB**

```
HKLM\System\CurrentControlSet\Services\NfsServer\Parameters\FileHandleCacheSizeinMB
```

The default is 4. This parameter specifies the maximum memory to be consumed by file handle cache entries. The minimum is 1 and the maximum is 1*1024*1024*1024 (1073741824).

- **LockFileHandleCacheInMemory**

```
HKLM\System\CurrentControlSet\Services\NfsServer\Parameters\LockFileHandleCacheInMemory
```

The default is 0. This parameter specifies whether the physical pages that are allocated for the cache size specified by FileHandleCacheSizeInMB are locked in memory. Setting this value to 1 enables this activity. Pages are locked in memory (not paged to disk), which improves the performance of resolving file handles, but reduces the memory that is available to applications.

- **MaxIcbNfsReadHandlesCacheSize**

```
HKLM\System\CurrentControlSet\Services\NfsServer\Parameters\MaxIcbNfsReadHandlesCacheSize
```

The default is 64. This parameter specifies the maximum number of handles per volume for the read data cache. Read cache entries are created only on systems that have more than 1 GB of memory. The minimum is 0 and the maximum is 0xFFFFFFFF.

- **HandleSigningEnabled**

```
HKLM\System\CurrentControlSet\Services\NfsServer\Parameters\HandleSigningEnabled
```

The default is 1. This parameter controls whether handles that are given out by NFS File Server are signed cryptographically. Setting it to 0 disables handle signing.

- **RdWrNfsDeferredWritesFlushDelay**

```
HKLM\System\CurrentControlSet\Services\NfsServer\Parameters\RdWrNfsDeferredWritesFlushDelay
```

The default is 60. This parameter is a soft timeout that controls the duration of NFS V3 UNSTABLE Write data caching. The minimum is 1, and the maximum is 600. Actual lifetime is approximately equal to RdWrNfsDeferredWritesFlushDelay multiplied by RdWrThreadSleepTime.

- **CacheAddFromCreateAndMkDir**

```
HKLM\System\CurrentControlSet\Services\NfsServer\Parameters\CacheAddFromCreateAndMkDir
```

The default is 1 (enabled). This parameter controls whether handles that are opened during NFS V2 and V3 CREATE and MKDIR RPC procedure handlers are retained in the file handle cache. Set this value to 0 to disable adding entries to the cache in CREATE and MKDIR code paths.

- **AdditionalDelayedWorkerThreads**

```
HKLM\SYSTEM\CurrentControlSet\Control\SessionManager\Executive\AdditionalDelayedWorkerThreads
```

Increases the number of delayed worker threads that are created for the specified work queue. Delayed worker threads process work items that are not considered time-critical and that can have their memory stack paged out while waiting for work items. An insufficient number of threads reduces the rate at which work items are serviced; a value that is too high consumes system resources unnecessarily.

- **NtfsDisable8dot3NameCreation**

```
HKLM\System\CurrentControlSet\Control\FileSystem\NtfsDisable8dot3NameCreation
```

The default in Windows Server 2012, Windows Server 2012 R2, and later versions of Windows Server is 2. In releases prior to Windows Server 2012, the default is 0. This parameter determines whether NTFS generates a short name in the 8dot3 (MSDOS) 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 8dot3 naming convention, NTFS does not generate a short name. A value of 2 means that this parameter can be configured per volume.

> **NOTE**
>
> The system volume has 8dot3 enabled by default. All other volumes in Windows Server 2012 and Windows Server 2012 R2 have 8dot3 disabled by default. 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 (disabled).

- **NtfsDisableLastAccessUpdate**

```
HKLM\System\CurrentControlSet\Control\FileSystem\NtfsDisableLastAccessUpdate
```

The default is 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.

- **MaxConcurrentConnectionsPerIp**

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Rpcxdr\Parameters\MaxConcurrentConnectionsPerIp
```

The default value of the MaxConcurrentConnectionsPerIp parameter is 16. You can increase this value up to a maximum of 8192 to increase the number of connections per IP address.

Performance Tuning Hyper-V Servers

12/16/2022 • 2 minutes to read • Edit Online

Hyper-V is the virtualization server role in Windows Server 2016. Virtualization servers can host multiple virtual machines that 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 energy efficiency and reduce the operational and maintenance costs of servers. In addition, virtual machines and the management APIs offer more flexibility for managing resources, balancing load, and provisioning systems.

Additional References

- Hyper-V terminology
- Hyper-V architecture
- Hyper-V server configuration
- Hyper-V processor performance
- Hyper-V memory performance
- Hyper-V storage I/O performance
- Hyper-V network I/O performance
- Detecting bottlenecks in a virtualized environment
- Linux Virtual Machines

Hyper-V Terminology

12/16/2022 • 2 minutes to read • Edit Online

This section summarizes key terminology specific to virtual machine technology that is used throughout this performance tuning topic:

TERM	DEFINITION
child partition	Any virtual machine 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 virtual machine environments and tune its behavior for virtual machines.
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 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 virtualized hardware resources (central processing unit or CPU, memory, and devices). The hypervisor controls and arbitrates access to the underlying hardware.
logical processor	A processing unit that handles one thread of execution (instruction stream). There can be one or more logical processors per processor core and one or more cores per processor socket.
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	The root partition that is created first and owns all the resources that the hypervisor does not, including most devices and system memory. The root partition hosts the virtualization stack and creates and manages the child partitions.

TERM	DEFINITION
Hyper-V-specific device	A virtualized device with no physical hardware analog, so guests may need a driver (virtualization service client) to that Hyper-V-specific device. The driver can use virtual machine bus (VMBus) to communicate with the virtualized device software in the root partition.
virtual machine	A virtual computer that was created by software emulation and has the same characteristics as a real computer.
virtual network switch	(also referred to as a virtual switch) A virtual version of a physical network switch. A virtual network can be configured to provide access to local or external network resources for one or more virtual machines.
virtual processor	A virtual abstraction of a processor that is scheduled to run on a logical processor. A virtual machine 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 by the virtualization stack in the root partition 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 virtual machines. The virtualization stack works with and sits above the hypervisor. It also provides management capabilities.
VMBus	Channel-based communication mechanism used for inter-partition communication and device enumeration on systems with multiple active virtualized partitions. The VMBus is installed with Hyper-V Integration Services.

Additional References

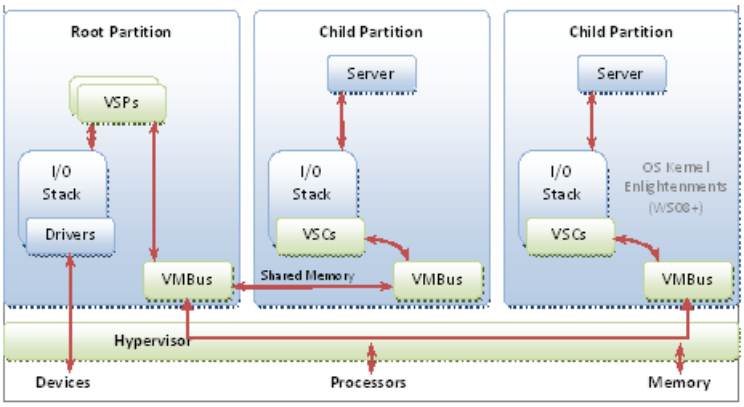
- Hyper-V architecture
- Hyper-V server configuration
- Hyper-V processor performance
- Hyper-V memory performance
- Hyper-V storage I/O performance
- Hyper-V network I/O performance
- Detecting bottlenecks in a virtualized environment
- Linux Virtual Machines

Hyper-V Architecture

12/16/2022 • 2 minutes to read • Edit Online

Hyper-V features a Type 1 hypervisor-based architecture. The hypervisor virtualizes processors and memory and provides mechanisms for the virtualization stack in the root partition to manage child partitions (virtual machines) and expose services such as I/O devices to the virtual machines.

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 virtual machines, management APIs, and virtualized I/O devices. It also implements emulated devices such as the integrated device electronics (IDE) disk controller and PS/2 input device port, and it supports Hyper-V-specific synthetic devices for increased performance and reduced overhead.



The Hyper-V-specific I/O architecture consists of virtualization service providers (VSPs) in the root partition and virtualization service clients (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 virtual machines that use mechanisms such as shared memory. The guest operating system's Plug and Play manager enumerates these devices, including VMBus, and loads the appropriate device drivers (virtual service clients). Services other than I/O are also exposed through this architecture.

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

The following sections suggest best practices that yield increased performance on servers running Hyper-V role.

Additional References

- Hyper-V terminology
- Hyper-V server configuration
- Hyper-V processor performance
- Hyper-V memory performance
- Hyper-V storage I/O performance
- Hyper-V network I/O performance
- Detecting bottlenecks in a virtualized environment
- Linux Virtual Machines

Hyper-V Configuration

12/16/2022 • 5 minutes to read • Edit Online

Hardware selection

The hardware considerations for servers running Hyper-V generally resemble those of non-virtualized servers, but servers running Hyper-V can exhibit increased CPU usage, consume more memory, and need larger I/O bandwidth because of server consolidation.

- **Processors**

  Hyper-V in Windows Server 2016 presents the logical processors as one or more virtual processors to each active virtual machine. Hyper-V now requires processors that support Second Level Address Translation (SLAT) technologies such as Extended Page Tables (EPT) or Nested Page Tables (NPT).

- **Cache**

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

- **Memory**

  The physical server requires sufficient memory for the both the root and child partitions. The root partition requires memory to efficiently perform I/Os on behalf of the virtual machines and operations such as a virtual machine snapshot. Hyper-V ensures that sufficient memory is available to the root partition, and allows remaining memory to be assigned to child partitions. Child partitions should be sized based on the needs of the expected load for each virtual machine.

- **Storage**

  The storage hardware should have sufficient I/O bandwidth and capacity to meet the current and future needs of the virtual machines that the physical server hosts. Consider these requirements when you select storage controllers and disks and choose the RAID configuration. Placing virtual machines with highly disk-intensive workloads on different physical disks will likely improve overall performance. For example, if four virtual machines share a single disk and actively use it, each virtual machine can yield only 25 percent of the bandwidth of that disk.

Power plan considerations

As a core technology, virtualization is a powerful tool useful in increasing server workload density, reducing the number of required physical servers in your datacenter, increasing operational efficiency and reducing power consumption costs. Power management is critical for cost management.

In an ideal datacenter environment, power consumption is managed by consolidating work onto machines until they're mostly busy and then turning off idle machines. If this approach is not practical, administrators can leverage power plans on the physical hosts to ensure they do not consume more power than necessary.

Server power management techniques come with a cost, particularly as tenant workloads are not trusted to dictate policy about the hoster's physical infrastructure. The host layer software is left to infer how to maximize throughput while minimizing power consumption. In mostly-idle machines, this can cause the physical infrastructure to conclude that moderate power draw is appropriate, resulting in individual tenant workloads running more slowly than they might otherwise.

Windows Server uses virtualization in a wide variety of scenarios. From a lightly loaded IIS Server to a moderately busy SQL Server, to a cloud host with Hyper-V running hundreds of virtual machines per server. Each of these scenarios may have unique hardware, software, and performance requirements. By default, Windows Server uses and recommends the **Balanced** power plan which enables power conservation by scaling the processor performance based on CPU utilization.

With the **Balanced** power plan, the highest power states (and lowest response latencies in tenant workloads) are applied only when the physical host is relatively busy. If you value deterministic, low-latency response for all tenant workloads, you should consider switching from the default **Balanced** power plan to the **High Performance** power plan. The **High Performance** power plan will run the processors at full speed all the time, effectively disabling Demand-Based Switching along with other power management techniques, and optimize for performance over power savings.

For customers, who are satisfied with the cost savings from reducing the number of physical servers and want to ensure they achieve maximum performance for their virtualized workloads, you should consider using the **High Performance** power plan.

Server Core installation option

Windows Server 2016 feature 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 footprint for the host OS, and a smaller attack and servicing surface. Therefore, we highly recommend that Hyper-V virtualization servers use the Server Core installation option.

A Server Core installation offers a console window only when the user is logged on, but Hyper-V exposes remote management features including Windows PowerShell so administrators can manage it remotely.

Dedicated server role

The root partition should be dedicated to Hyper-V. Running additional server roles on a server running Hyper-V 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.

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

Guest operating systems

Hyper-V supports and has been tuned for a number of different guest operating systems. The number of virtual processors that are supported per guest depends on the guest operating system. For a list of the supported guest operating systems, see Hyper-V Overview.

CPU statistics

Hyper-V publishes performance counters to help characterize the behavior of the virtualization server and report the resource usage. The standard set of tools for viewing performance counters in Windows includes Performance Monitor 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 by using the Hyper-V Hypervisor Logical Processor performance counters. The CPU utilization counters that Task Manager and Performance Monitor report in the root and child partitions do not reflect the actual physical CPU usage. Use the following performance counters to monitor performance:

- **Hyper-V Hypervisor Logical Processor (*)\% Total Run Time** The total non-idle time of the logical processors
- **Hyper-V Hypervisor Logical Processor (*)\% Guest Run Time** The time spent running cycles within a guest or within the host
- **Hyper-V Hypervisor Logical Processor (*)\% Hypervisor Run Time** The time spent running within the hypervisor
- **Hyper-V Hypervisor Root Virtual Processor (*)*** Measures the CPU usage of the root partition
- **Hyper-V Hypervisor Virtual Processor (*)*** Measures the CPU usage of guest partitions

Additional References

- Hyper-V terminology
- Hyper-V architecture
- Hyper-V processor performance
- Hyper-V memory performance
- Hyper-V storage I/O performance
- Hyper-V network I/O performance
- Detecting bottlenecks in a virtualized environment
- Linux Virtual Machines

Hyper-V Processor Performance

12/16/2022 • 3 minutes to read • Edit Online

Virtual machine integration services

The Virtual Machine Integration Services include enlightened drivers for the Hyper-V-specific I/O devices, which significantly reduces CPU overhead for I/O compared to emulated devices. You should install the latest version of the Virtual Machine Integration Services in every supported virtual machine. The services decrease the CPU usage of the guests, from idle guests to heavily used guests, and improves the I/O throughput. This is the first step in tuning performance in a server running Hyper-V. For a list of supported guest operating systems, see Hyper-V Overview.

Virtual processors

Hyper-V in Windows Server 2016 supports a maximum of 240 virtual processors per virtual machine. Virtual machines that have loads that are not CPU intensive should be configured to use 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.

Increase the number of virtual processors if the virtual machine requires more than one CPU of processing under peak load.

Background activity

Minimizing the background activity in idle virtual machines releases CPU cycles that can be used elsewhere by other virtual machines. Windows guests typically use less than one percent of one CPU when they are idle. The following are several best practices for minimizing the background CPU usage of a virtual machine:

- Install the latest version of the Virtual Machine Integration Services.
- Remove the emulated network adapter through the virtual machine settings dialog box (use the Microsoft Hyper-V-specific adapter).
- Remove unused devices such as the CD-ROM and COM port, or disconnect their media.
- Keep the Windows guest operating system on the sign-in screen when it is not being used and disable the screen saver.
- Review the scheduled tasks and services that are enabled by default.
- Review the ETW trace providers that are on by default by running **logman.exe query -ets**
- Improve server applications to reduce periodic activity (such as timers).
- Close Server Manager on both the host and guest operating systems.
- Don't leave Hyper-V Manager running since it constantly refreshes the virtual machine's thumbnail.

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

- Disable background services such as SuperFetch and Windows Search.
- Disable scheduled tasks such as Scheduled Defrag.

Virtual NUMA

To enable virtualizing large scale-up workloads, Hyper-V in Windows Server 2016 expanded virtual machine scale limits. A single virtual machine can be assigned up to 240 virtual processors and 12 TB of memory. When creating such large virtual machines, memory from multiple NUMA nodes on the host system will likely be utilized. In such virtual machine configuration, if virtual processors and memory are not allocated from the same NUMA node, workloads may have bad performance due to the inability to take advantage of NUMA optimizations.

In Windows Server 2016, Hyper-V presents a virtual NUMA topology to virtual machines. By default, this virtual NUMA topology is optimized to match the NUMA topology of the underlying host computer. Exposing a virtual NUMA topology into a virtual machine allows the guest operating system and any NUMA-aware applications running within it to take advantage of the NUMA performance optimizations, just as they would when running on a physical computer.

There is no distinction between a virtual and a physical NUMA from the workload's perspective. Inside a virtual machine, when a workload allocates local memory for data, and accesses that data in the same NUMA node, fast local memory access results on the underlying physical system. Performance penalties due to remote memory access are successfully avoided. Only NUMA-aware applications can benefit of vNUMA.

Microsoft SQL Server is an example of NUMA aware application. For more info, see Understanding Non-uniform Memory Access.

Virtual NUMA and Dynamic Memory features cannot be used at the same time. A virtual machine that has Dynamic Memory enabled effectively has only one virtual NUMA node, and no NUMA topology is presented to the virtual machine regardless of the virtual NUMA settings.

For more info on Virtual NUMA, see Hyper-V Virtual NUMA Overview.

Additional References

- Hyper-V terminology
- Hyper-V architecture
- Hyper-V server configuration
- Hyper-V memory performance
- Hyper-V storage I/O performance
- Hyper-V network I/O performance
- Detecting bottlenecks in a virtualized environment
- Linux Virtual Machines

Hyper-V Memory Performance

12/16/2022 • 2 minutes to read • Edit Online

The hypervisor virtualizes the guest physical memory to isolate virtual machines from each other and to provide a contiguous, zero-based memory space for each guest operating system, just as on non-virtualized systems.

Correct memory sizing for child partitions

You should size virtual machine memory as you typically do for server applications on a physical computer. 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.

You can enable Dynamic Memory to allow Windows to size virtual machine memory dynamically. With Dynamic Memory, if applications in the virtual machine experience problems making large sudden memory allocations, you can increase the page file size for the virtual machine to ensure temporary backing while Dynamic Memory responds to the memory pressure.

For more info on Dynamic Memory, see Hyper-V Dynamic Memory Overview and Hyper-V Dynamic Memory Configuration Guide.

When running Windows in the child partition, you can use the following performance counters within a child partition to identify whether the child partition is experiencing memory pressure and is likely to perform better with a higher virtual machine memory size.

PERFORMANCE COUNTER	SUGGESTED THRESHOLD VALUE
Memory – Standby Cache Reserve Bytes	Sum of Standby Cache Reserve Bytes and Free and Zero Page List Bytes should be 200 MB or more on systems with 1 GB, and 300 MB or more on systems with 2 GB or more of visible RAM.
Memory – Free & Zero Page List Bytes	Sum of Standby Cache Reserve Bytes and Free and Zero Page List Bytes should be 200 MB or more on systems with 1 GB, and 300 MB or more on systems with 2 GB or more of visible RAM.
Memory – Pages Input/Sec	Average over a 1-hour period is less than 10.

Correct memory sizing for root partition

The root partition must have sufficient memory to provide services such as I/O virtualization, virtual machine snapshot, and management to support the child partitions.

Hyper-V in Windows Server 2016 monitors the runtime health of the root partition's management operating system to determine how much memory can safely be allocated to child partitions, while still ensuring high performance and reliability of the root partition.

Additional References

- Hyper-V terminology

- Hyper-V architecture
- Hyper-V server configuration
- Hyper-V processor performance
- Hyper-V storage I/O performance
- Hyper-V network I/O performance
- Detecting bottlenecks in a virtualized environment
- Linux Virtual Machines

Hyper-V Storage I/O Performance

12/16/2022 • 17 minutes to read • Edit Online

This section describes the different options and considerations for tuning storage I/O performance in a virtual machine. The storage I/O path extends from the guest storage stack, through the host virtualization layer, to the host storage stack, and then to the physical disk. Following are explanations about how optimizations are possible at each of these stages.

Virtual controllers

Hyper-V offers three types of virtual controllers: IDE, SCSI, and Virtual host bus adapters (HBAs).

IDE

IDE controllers expose IDE disks to the virtual machine. The IDE controller is emulated, and it is the only controller that is available for guest VMs running older version of Windows without the Virtual Machine Integration Services. Disk I/O that is performed by using the IDE filter driver that is provided with the Virtual Machine Integration Services is significantly better than the disk I/O performance that is provided with the emulated IDE controller. We recommend that IDE disks be used only for the operating system disks because they have performance limitations due to the maximum I/O size that can be issued to these devices.

SCSI (SAS controller)

SCSI controllers expose SCSI disks to the virtual machine, and each virtual SCSI controller can support up to 64 devices. For optimal performance, we recommend that you attach multiple disks to a single virtual SCSI controller and create additional controllers only as they are required to scale the number of disks connected to the virtual machine. SCSI path is not emulated which makes it the preferred controller for any disk other than the operating system disk. In fact with Generation 2 VMs, it is the only type of controller possible. Introduced in Windows Server 2012 R2, this controller is reported as SAS to support shared VHDX.

Virtual Fibre Channel HBAs

Virtual Fibre Channel HBAs can be configured to allow direct access for virtual machines to Fibre Channel and Fibre Channel over Ethernet (FCoE) LUNs. Virtual Fibre Channel disks bypass the NTFS file system in the root partition, which reduces the CPU usage of storage I/O.

Large data drives and drives that are shared between multiple virtual machines (for guest clustering scenarios) are prime candidates for virtual Fibre Channel disks.

Virtual Fibre Channel disks require one or more Fibre Channel host bus adapters (HBAs) to be installed on the host. Each host HBA is required to use an HBA driver that supports the Windows Server 2016 Virtual Fibre Channel/NPIV capabilities. The SAN fabric should support NPIV, and the HBA port(s) that are used for the virtual Fibre Channel should be set up in a Fibre Channel topology that supports NPIV.

To maximize throughput on hosts that are installed with more than one HBA, we recommend that you configure multiple virtual HBAs inside the Hyper-V virtual machine (up to four HBAs can be configured for each virtual machine). Hyper-V will automatically make a best effort to balance virtual HBAs to host HBAs that access the same virtual SAN.

Virtual disks

Disks can be exposed to the virtual machines through the virtual controllers. These disks could be virtual hard disks that are file abstractions of a disk or a pass-through disk on the host.

Virtual hard disks

There are two virtual hard disk formats, VHD and VHDX. Each of these formats supports three types of virtual hard disk files.

VHD format

The VHD format was the only virtual hard disk format that was supported by Hyper-V in past releases. Introduced in Windows Server 2012, the VHD format has been modified to allow better alignment, which results in significantly better performance on new large sector disks.

Any new VHD that is created on a Windows Server 2012 or newer has the optimal 4 KB alignment. This aligned format is completely compatible with previous Windows Server operating systems. However, the alignment property will be broken for new allocations from parsers that are not 4 KB alignment-aware (such as a VHD parser from a previous version of Windows Server or a non-Microsoft parser).

Any VHD that is moved from a previous release does not automatically get converted to this new improved VHD format.

To convert to new VHD format, run the following Windows PowerShell command:

```
Convert-VHD –Path E:\vms\testvhd\test.vhd –DestinationPath E:\vms\testvhd\test-converted.vhd
```

You can check the alignment property for all the VHDs on the system, and it should be converted to the optimal 4 KB alignment. You create a new VHD with the data from the original VHD by using the **Create-from-Source** option.

To check for alignment by using Windows Powershell, examine the Alignment line, as shown below:

```
Get-VHD –Path E:\vms\testvhd\test.vhd

Path                    : E:\vms\testvhd\test.vhd
VhdFormat               : VHD
VhdType                 : Dynamic
FileSize                : 69245440
Size                    : 10737418240
MinimumSize             : 10735321088
LogicalSectorSize       : 512
PhysicalSectorSize      : 512
BlockSize               : 2097152
ParentPath              :
FragmentationPercentage : 10
Alignment               : 0
Attached                : False
DiskNumber              :
IsDeleted               : False
Number                  :
```

To verify alignment by using Windows PowerShell, examine the Alignment line, as shown below:

```
Get-VHD –Path E:\vms\testvhd\test-converted.vhd

Path                    : E:\vms\testvhd\test-converted.vhd
VhdFormat               : VHD
VhdType                 : Dynamic
FileSize                : 69369856
Size                    : 10737418240
MinimumSize             : 10735321088
LogicalSectorSize       : 512
PhysicalSectorSize      : 512
BlockSize               : 2097152
ParentPath              :
FragmentationPercentage : 0
Alignment               : 1
Attached                : False
DiskNumber              :
IsDeleted               : False
Number                  :
```

VHDX format

VHDX is a new virtual hard disk format introduced in Windows Server 2012, which allows you to create resilient high-performance virtual disks up to 64 terabytes. Benefits of this format include:

- Support for virtual hard disk storage capacity of up to 64 terabytes.
- Protection against data corruption during power failures by logging updates to the VHDX metadata structures.
- Ability to store custom metadata about a file, which a user might want to record, such as operating system version or patches applied.

The VHDX format also provides the following performance benefits:

- Improved alignment of the virtual hard disk format to work well on large sector disks.
- Larger block sizes for dynamic and differential disks, which allows these disks to attune to the needs of the workload.
- 4 KB logical sector virtual disk that allows for increased performance when used by applications and workloads that are designed for 4 KB sectors.
- Efficiency in representing data, which results in smaller file size and allows the underlying physical storage device to reclaim unused space. (Trim requires pass-through or SCSI disks and trim-compatible hardware.)

When you upgrade to Windows Server 2016, we recommend that you convert all VHD files to the VHDX format due to these benefits. The only scenario where it would make sense to keep the files in the VHD format is when a virtual machine has the potential to be moved to a previous release of Hyper-V that does not support the VHDX format.

Types of virtual hard disk files

There are three types of VHD files. The following sections are the performance characteristics and trade-offs between the types.

The following recommendations should be taken into consideration with regards to selecting a VHD file type:

- When using the VHD format, we recommend that you use the fixed type because it has better resiliency and performance characteristics compared to the other VHD file types.

- When using the VHDX format, we recommend that you use the dynamic type because it offers resiliency guarantees in addition to space savings that are associated with allocating space only when there is a need to do so.
- The fixed type is also recommended, irrespective of the format, when the storage on the hosting volume is not actively monitored to ensure that sufficient disk space is present when expanding the VHD file at run time.
- Snapshots of a virtual machine create a differencing VHD to store writes to the disks. 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.

Fixed virtual hard disk type

Space for the VHD is first allocated when the VHD file is created. This type of VHD file is less likely 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 file types because reads and writes do not need to look up the mapping of the block.

Dynamic virtual hard disk type

Space for the VHD is allocated on demand. The blocks in the disk start as unallocated blocks and are not backed by any actual space in the file. 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 increases CPU usage. Reads and writes to existing blocks incur disk access and CPU overhead when looking up the blocks' mapping in the metadata.

Differencing virtual hard disk type

The VHD points to a parent VHD file. Any writes to blocks not written to 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 VHD file.

Block size considerations

Block size can significantly impact performance. It is optimal to match the block size to the allocation patterns of the workload that is using the disk. For example, if an application is allocating in chunks of 16 MB, it would be optimal to have a virtual hard disk block size of 16 MB. A block size of >2 MB is possible only on virtual hard disks with the VHDX format. Having a larger block size than the allocation pattern for a random I/O workload will significantly increase the space usage on the host.

Sector size implications

Most of the software industry has depended on disk sectors of 512 bytes, but the standard is moving to 4 KB disk sectors. To reduce compatibility issues that might arise from a change in sector size, hard drive vendors are introducing a transitional size referred to as 512 emulation drives (512e).

These emulation drives offer some of the advantages that are offered by 4 KB disk sector native drives, such as improved format efficiency and an improved scheme for error correction codes (ECC). They come with fewer compatibility issues that would occur by exposing a 4 KB sector size at the disk interface.

Support for 512e disks

A 512e disk can perform a write only in terms of a physical sector—that is, it cannot directly write a 512byte sector that is issued to it. The internal process in the disk that makes these writes possible follows these steps:

- The disk reads the 4 KB physical sector to its internal cache, which contains the 512-byte logical sector referred to in the write.
- Data in the 4 KB buffer is modified to include the updated 512-byte sector.
- The disk performs a write of the updated 4 KB buffer back to its physical sector on the disk.

This process is called read-modify-write (RMW). The overall performance impact of the RMW process depends on the workloads. The RMW process causes performance degradation in virtual hard disks for the following reasons:

- Dynamic and differencing virtual hard disks have a 512-byte sector bitmap in front of their data payload. In addition, footer, header, and parent locators align to a 512-byte sector. It is common for the virtual hard disk driver to issue 512-byte write commands to update these structures, resulting in the RMW process described earlier.
- Applications commonly issue reads and writes in multiples of 4 KB sizes (the default cluster size of NTFS). Because there is a 512-byte sector bitmap in front of the data payload block of dynamic and differencing virtual hard disks, the 4 KB blocks are not aligned to the physical 4 KB boundary. The following figure shows a VHD 4 KB block (highlighted) that is not aligned with physical 4 KB boundary.

Each 4 KB write command that is issued by the current parser to update the payload data results in two reads for two blocks on the disk, which are then updated and subsequently written back to the two disk blocks. Hyper-V in Windows Server 2016 mitigates some of the performance effects on 512e disks on the VHD stack by preparing the previously mentioned structures for alignment to 4 KB boundaries in the VHD format. This avoids the RMW effect when accessing the data within the virtual hard disk file and when updating the virtual hard disk metadata structures.

As mentioned earlier, VHDs that are copied from previous versions of Windows Server will not automatically be aligned to 4 KB. You can manually convert them to optimally align by using the **Copy from Source** disk option that is available in the VHD interfaces.

By default, VHDs are exposed with a physical sector size of 512 bytes. This is done to ensure that physical sector size dependent applications are not impacted when the application and VHDs are moved from a previous version of Windows Server.

By default, disks with the VHDX format are created with the 4 KB physical sector size to optimize their performance profile regular disks and large sector disks. To make full use of 4 KB sectors it's recommended to use VHDX format.

Support for native 4 KB disks

Hyper-V in Windows Server 2012 R2 and beyond supports 4 KB native disks. But it is still possible to store VHD disk on 4 KB native disk. This is done by implementing a software RMW algorithm in the virtual storage stack layer that converts 512-byte access and update requests to corresponding 4 KB accesses and updates.

Because VHD file can only expose themselves as 512-byte logical sector size disks, it is very likely that there will be applications that issue 512-byte I/O requests. In these cases, the RMW layer will satisfy these requests and cause performance degradation. This is also true for a disk that is formatted with VHDX that has a logical sector size of 512 bytes.

It is possible to configure a VHDX file to be exposed as a 4 KB logical sector size disk, and this would be an optimal configuration for performance when the disk is hosted on a 4 KB native physical device. Care should be taken to ensure that the guest and the application that is using the virtual disk are backed by the 4 KB logical sector size. The VHDX formatting will work correctly on a 4 KB logical sector size device.

Pass-through disks

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

Pass-through disks should be avoided due to the limitations introduced with virtual machine migration scenarios.

Advanced storage features

Storage Quality of Service (QoS)

Starting in Windows Server 2012 R2, Hyper-V includes the ability to set certain quality-of-service (QoS) parameters for storage on the virtual machines. Storage QoS provides storage performance isolation in a multitenant environment and mechanisms to notify you when the storage I/O performance does not meet the defined threshold to efficiently run your virtual machine workloads.

Storage QoS provides the ability to specify a maximum input/output operations per second (IOPS) value for your virtual hard disk. An administrator can throttle the storage I/O to stop a tenant from consuming excessive storage resources that may impact another tenant.

You can also set a minimum IOPS value. They will be notified when the IOPS to a specified virtual hard disk is below a threshold that is needed for its optimal performance.

The virtual machine metrics infrastructure is also updated, with storage related parameters to allow the administrator to monitor the performance and chargeback related parameters.

Maximum and minimum values are specified in terms of normalized IOPS where every 8 KB of data is counted as an I/O.

Some of the limitations are as follows:

- Only for virtual disks
- Differencing disk cannot have parent virtual disk on a different volume
- Replica - QoS for replica site configured separately from primary site
- Shared VHDX is not supported

For more info on Storage Quality of Service, see Storage Quality of Service for Hyper-V.

NUMA I/O

Windows Server 2012 and beyond supports large virtual machines, and any large virtual machine configuration (for example, a configuration with Microsoft SQL Server running with 64 virtual processors) will also need scalability in terms of I/O throughput.

The following key improvements first introduced in the Windows Server 2012 storage stack and Hyper-V provide the I/O scalability needs of large virtual machines:

- An increase in the number of communication channels created between the guest devices and host storage stack.
- A more efficient I/O completion mechanism involving interrupt distribution amongst the virtual processors to avoid expensive interprocessor interruptions.

Introduced in Windows Server 2012, there are a few registry entries, located at HKLM\System\CurrentControlSet\Enum\VMBUS\{device id}\{instance id}\StorChannel, that allow the number of channels to be adjusted. They also align the virtual processors that handle the I/O completions to the virtual CPUs that are assigned by the application to be the I/O processors. The registry settings are configured on a per-adapter basis on the device's hardware key.

- **ChannelCount (DWORD)** The total number of channels to use, with a maximum of 16. It defaults to a ceiling, which is the number of virtual processors/16.
- **ChannelMask (QWORD)** The processor affinity for the channels. If it is not set or is set to 0, it defaults to the existing channel distribution algorithm that you use for normal storage or for networking channels. This ensures that your storage channels won't conflict with your network channels.

Offloaded Data Transfer integration

Crucial maintenance tasks for VHDs, such as merge, move, and compact, depend copying large amounts of data. The current method of copying data requires data to be read in and written to different locations, which can be a time-consuming process. It also uses CPU and memory resources on the host, which could have been used to service virtual machines.

Storage area network (SAN) vendors are working to provide near-instantaneous copy operations of large amounts of data. This storage is designed to allow the system above the disks to specify the move of a specific data set from one location to another. This hardware feature is known as an Offloaded Data Transfer.

Hyper-V in Windows Server 2012 and beyond supports Offload Data Transfer (ODX) operations so that these operations can be passed from the guest operating system to the host hardware. This ensures that the workload can use ODX-enabled storage as it would if it were running in a non-virtualized environment. The Hyper-V storage stack also issues ODX operations during maintenance operations for VHDs such as merging disks and storage migration meta-operations where large amounts of data are moved.

Unmap integration

Virtual hard disk files exist as files on a storage volume, and they share available space with other files. Because the size of these files tends to be large, the space that they consume can grow quickly. Demand for more physical storage affects the IT hardware budget. It's important to optimize the use of physical storage as much as possible.

Before Windows Server 2012, when applications delete content within a virtual hard disk, which effectively abandoned the content's storage space, the Windows storage stack in the guest operating system and the Hyper-V host had limitations that prevented this information from being communicated to the virtual hard disk and the physical storage device. This prevented the Hyper-V storage stack from optimizing the space usage by the VHD-based virtual disk files. It also prevented the underlying storage device from reclaiming the space that was previously occupied by the deleted data.

Starting from Windows Server 2012, Hyper-V supports unmap notifications, which allow VHDX files to be more efficient in representing that data within it. This results in smaller files size, and it allows the underlying physical storage device to reclaim unused space.

Only Hyper-V-specific SCSI, enlightened IDE, and Virtual Fibre Channel controllers allow the unmap command from the guest to reach the host virtual storage stack. On the virtual hard disks, only virtual disks formatted as

VHDX support unmap commands from the guest.

For these reasons, we recommend that you use VHDX files attached to a SCSI controller when not using Virtual Fibre Channel disks.

Additional References

- Hyper-V terminology
- Hyper-V architecture
- Hyper-V server configuration
- Hyper-V processor performance
- Hyper-V memory performance
- Hyper-V network I/O performance
- Detecting bottlenecks in a virtualized environment
- Linux Virtual Machines

Hyper-V Network I/O Performance

12/16/2022 • 2 minutes to read • Edit Online

Server 2016 contains several improvements and new functionality to optimize network performance under Hyper-V. Documentation on how to optimize network performance will be included in a future version of this article.

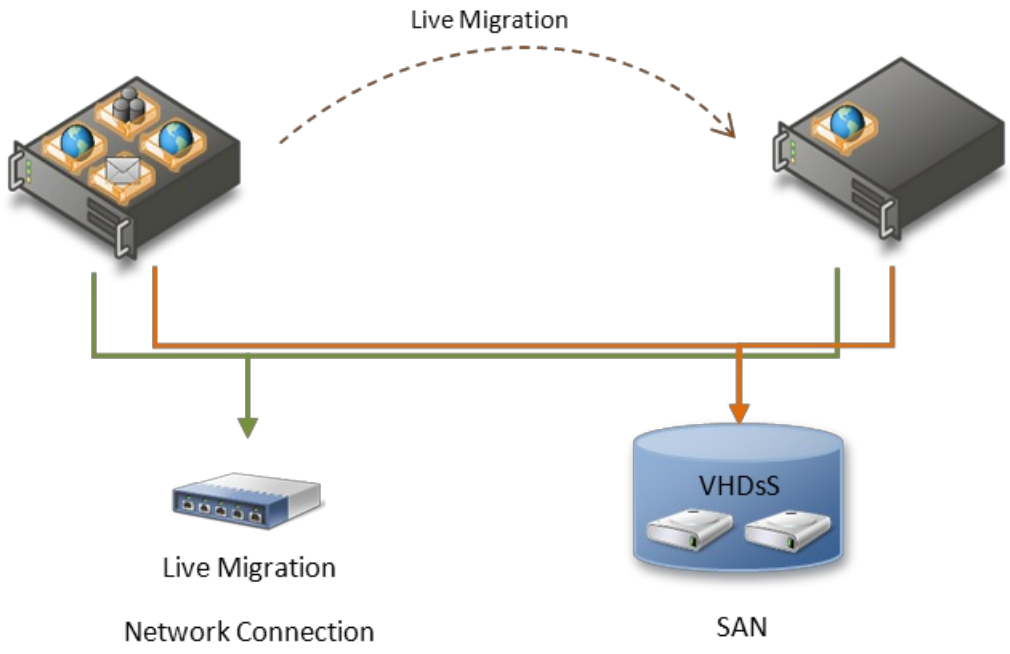
Live Migration

Live Migration lets you to transparently move running virtual machines from one node of a failover cluster to another node in the same cluster without a dropped network connection or perceived downtime.

> **NOTE**
> Failover Clustering requires shared storage for the cluster nodes.

The process of moving a running virtual machine can be divided into two major phases. The first phase copies the memory of the virtual machine from the current host to the new host. The second phase transfers the virtual machine state from the current host to the new host. The durations of both phases is greatly determined by the speed at which data can be transferred from the current host to the new host.

Providing a dedicated network for live migration traffic helps minimize the time that is required to complete a live migration, and it ensures consistent migration times.



Additionally, increasing the number of send and receive buffers on each network adapter that is involved in the migration can improve migration performance.

Windows Server 2012 R2 introduced an option to speed up Live Migration by compressing memory before transferring over the network or use Remote Direct Memory Access (RDMA), if your hardware supports it.

Additional References

- Hyper-V terminology
- Hyper-V architecture
- Hyper-V server configuration
- Hyper-V processor performance
- Hyper-V memory performance
- Hyper-V storage I/O performance
- Detecting bottlenecks in a virtualized environment
- Linux Virtual Machines

Detecting bottlenecks in a virtualized environment

12/16/2022 • 3 minutes to read • Edit Online

This section should give you some hints on what to monitor by using Performance Monitor and how to identify where the problem might be when either the host or some of the virtual machines do not perform as you would have expected.

Processor bottlenecks

Here are some common scenarios that could cause processor bottlenecks:

- One or more logical processors are loaded
- One or more virtual processors are loaded

You can use the following performance counters from the host:

- Logical Processor Utilization - \Hyper-V Hypervisor Logical Processor(*)\% Total Run Time
- Virtual Processor Utilization - \Hyper-V Hypervisor Virtual Processor(*)\% Total Run Time
- Root Virtual Processor Utilization - \Hyper-V Hypervisor Root Virtual Processor(*)\% Total Run Time

If the **Hyper-V Hypervisor Logical Processor(_Total)\% Total Runtime** counter is over 90%, the host is overloaded. You should add more processing power or move some virtual machines to a different host.

If the **Hyper-V Hypervisor Virtual Processor(VM Name:VP x)\% Total Runtime** counter is over 90% for all virtual processors, you should do the following:

- Verify that the host is not overloaded
- Find out if the workload can leverage more virtual processors
- Assign more virtual processors to the virtual machine

If **Hyper-V Hypervisor Virtual Processor(VM Name:VP x)\% Total Runtime** counter is over 90% for some, but not all, of the virtual processors, you should do the following:

- If your workload is receive network-intensive, you should consider using vRSS.
- If the virtual machines are not running Windows Server 2012 R2, you should add more network adapters.
- If your workload is storage-intensive, you should enable virtual NUMA and add more virtual disks.

If the **Hyper-V Hypervisor Root Virtual Processor (Root VP x)\% Total Runtime** counter is over 90% for some, but not all, virtual processors and the **Processor (x)\% Interrupt Time and Processor (x)\% DPC Time** counter approximately adds up to the value for the **Root Virtual Processor(Root VP x)\% Total Runtime** counter, you should ensure enable VMQ on the network adapters.

Memory bottlenecks

Here are some common scenarios that could cause memory bottlenecks:

- The host is not responsive.

- Virtual machines cannot be started.
- Virtual machines run out of memory.

You can use the following performance counters from the host:

- Memory\Available Mbytes
- Hyper-V Dynamic Memory Balancer (*)\Available Memory

You can use the following performance counters from the virtual machine:

- Memory\Available Mbytes

If the **Memory\Available Mbytes** and **Hyper-V Dynamic Memory Balancer (*)\Available Memory** counters are low on the host, you should stop non-essential services and migrate one or more virtual machines to another host.

If the **Memory\Available Mbytes** counter is low in the virtual machine, you should assign more memory to the virtual machine. If you are using Dynamic Memory, you should increase the maximum memory setting.

Network bottlenecks

Here are some common scenarios that could cause network bottlenecks:

- The host is network bound.
- The virtual machine is network bound.

You can use the following performance counters from the host:

- Network Interface(*network adapter name*)\Bytes/sec

You can use the following performance counters from the virtual machine:

- Hyper-V Virtual Network Adapter (*virtual machine name name<GUID>*)\Bytes/sec

If the **Physical NIC Bytes/sec** counter is greater than or equal to 90% of capacity, you should add additional network adapters, migrate virtual machines to another host, and configure Network QoS.

If the **Hyper-V Virtual Network Adapter Bytes/sec** counter is greater than or equal to 250 MBps, you should add additional teamed network adapters in the virtual machine, enable vRSS, and use SR-IOV.

If your workloads can't meet their network latency, enable SR-IOV to present physical network adapter resources to the virtual machine.

Storage bottlenecks

Here are some common scenarios that could cause storage bottlenecks:

- The host and virtual machine operations are slow or time out.
- The virtual machine is sluggish.

You can use the following performance counters from the host:

- Physical Disk(*disk letter*)\Avg. disk sec/Read
- Physical Disk(*disk letter*)\Avg. disk sec/Write
- Physical Disk(*disk letter*)\Avg. disk read queue length

- Physical Disk(*disk letter*)\Avg. disk write queue length

If latencies are consistently greater than 50ms, you should do the following:

- Spread virtual machines across additional storage
- Consider purchasing faster storage
- Consider Tiered Storage Spaces, which was introduced in Windows Server 2012 R2
- Consider using Storage QoS, which was introduced in Windows Server 2012 R2
- Use VHDX

Additional References

- Hyper-V terminology
- Hyper-V architecture
- Hyper-V server configuration
- Hyper-V processor performance
- Hyper-V memory performance
- Hyper-V storage I/O performance
- Hyper-V network I/O performance
- Linux Virtual Machines

Linux Virtual Machine Considerations

12/16/2022 • 2 minutes to read • Edit Online

Linux and BSD virtual machines have additional considerations compared to Windows virtual machines in Hyper-V.

The first consideration is whether Integration Services are present or if the VM is running merely on emulated hardware with no enlightenment. A table of Linux and BSD releases that have built-in or downloadable Integration Services is available in Supported Linux and FreeBSD virtual machines for Hyper-V on Windows. These pages have grids of available Hyper-V features available to Linux distribution releases, and notes on those features where applicable.

Even when the guest is running Integration Services, it can be configured with legacy hardware which does not exhibit the best performance. For example, configure and use a virtual ethernet adapter for the guest instead of using a legacy network adapter. With Windows Server 2016, advanced networking like SR-IOV are available as well.

Linux Network Performance

Linux by default enables hardware acceleration and offloads by default. If vRSS is enabled in the properties of a NIC on the host and the Linux guest has the capability to use vRSS the capability will be enabled. In Powershell this same parameter can be changed with the `EnableNetAdapterRSS` command.

Similarly, the VMMQ (Virtual Switch RSS) feature can be enabled on the physical NIC used by the guest **Properties** > **Configure...** > **Advanced** tab > set **Virtual Switch RSS** to **Enabled** or enable VMMQ in Powershell using the following:

```
Set-VMNetworkAdapter -VMName **$VMName** -VmmqEnabled $True
```

In the guest additional TCP tuning can be performed by increasing limits. For the best performance spreading workload over multiple CPUs and having deep workloads produces the best throughput, as virtualized workloads will have higher latency than "bare metal" ones.

Some example tuning parameters that have been useful in network benchmarks include:

```
net.core.netdev_max_backlog = 30000
net.core.rmem_max = 67108864
net.core.wmem_max = 67108864
net.ipv4.tcp_wmem = 4096 12582912 33554432
net.ipv4.tcp_rmem = 4096 12582912 33554432
net.ipv4.tcp_max_syn_backlog = 80960
net.ipv4.tcp_slow_start_after_idle = 0
net.ipv4.tcp_tw_reuse = 1
net.ipv4.ip_local_port_range = 10240 65535
net.ipv4.tcp_abort_on_overflow = 1
```

A useful tool for network microbenchmarks is ntttcp, which is available on both Linux and Windows. The Linux version is open source and available from ntttcp-for-linux on github.com. The Windows version can be found in the download center. When tuning workloads it is best to use as many streams as necessary to get the best throughput. Using ntttcp to model traffic, the `-P` parameter sets the number of parallel connections used.

Linux Storage Performance

Some best practices, like the following, are listed on Best Practices for Running Linux on Hyper-V. The Linux kernel has different I/O schedulers to reorder requests with different algorithms. NOOP is a first-in first-out queue that passes the schedule decision to be made by the hypervisor. It is recommended to use NOOP as the scheduler when running Linux virtual machine on Hyper-V. To change the scheduler for a specific device, in the boot loader's configuration (/etc/grub.conf, for example), add `elevator=noop` to the kernel parameters, and then restart.

Similar to networking, Linux guest performance with storage benefits the most from multiple queues with enough depth to keep the host busy. Microbenchmarking storage performance is probably best with the fio benchmark tool with the libaio engine.

Additional References

- Hyper-V terminology
- Hyper-V architecture
- Hyper-V server configuration
- Hyper-V processor performance
- Hyper-V memory performance
- Hyper-V storage I/O performance
- Hyper-V network I/O performance
- Detecting bottlenecks in a virtualized environment

Performance tuning Windows Server Containers

12/16/2022 • 5 minutes to read • Edit Online

Introduction

Starting with Windows Server 2022, two types of containers are available: Windows Server Containers and Hyper-V Containers. Each container type supports either the Server Core or Nano Server SKU of Windows Server 2022.

These configurations have different performance implications, which we detail below to help you understand which is right for your scenarios. In addition, we detail performance impacting configurations, and describe the tradeoffs with each of those options.

Windows Server Container and Hyper-V Containers

Windows Server Container and Hyper-V containers offer many of the same portability and consistency benefits but differ in terms of their isolation guarantees and performance characteristics.

Windows Server Containers provide application isolation through process and namespace isolation technology. A Windows Server container shares a kernel with the container host and all containers running on the host.

Hyper-V Containers expand on the isolation provided by Windows Server Containers by running each container in a highly optimized virtual machine. In this configuration, the kernel of the container host is not shared with the Hyper-V Containers.

The extra isolation provided by Hyper-V containers is achieved in large part by a hypervisor layer of isolation between the container and the container host. This affects container density as, unlike Windows Server Containers, less sharing of system files and binaries can occur, resulting in an overall larger storage and memory footprint. In addition, there is the expected further overhead in some network, storage IO, and CPU paths.

Nano Server and Server Core

Windows Server Containers and Hyper-V Containers offer support for Server Core, learn more about the container base image options.

Nano Server is a remotely administered server operating system optimized for private clouds and datacenters. It is similar to Windows Server in Server Core mode, but significantly smaller, has no local logon capability, and only supports 64-bit applications, tools, and agents. It takes up far less disk space, sets up significantly faster, and requires far fewer updates and restarts than Windows Server. When it does restart, it restarts much faster.

Container Start-Up Time

Container start-up time is a key metric in many of the scenarios that containers offer the greatest benefit. As such, understanding how to best optimize for container start-up time is critical. Below are some tuning trade-offs to understand to achieve improved start-up time.

First Logon

Microsoft ships a base image for both Nano Server and Server Core. The base image that ships for Server Core has been optimized by removing the start-up time overhead associated with first logon (OOBE). This is not the case with Nano Server base image. However, this cost can be removed from Nano Server based images by committing at least one layer to the container image. Subsequent container starts from the image will not incur the first logon cost.

Scratch Space Location

Containers, by default, use a temporary scratch space on the container host's system drive media for storage during the lifetime of the running container. This serves as the container's system drive, and as such many of the writes and reads done in container operation follow this path. For host systems where the system drive exists on spinning disk magnetic media (HDDs) but faster storage media is available (faster HDDs or SSDs), it is possible to move the container scratch space to a different drive. This is achieved by using the dockerd –g command. This command is global, and will affect all containers running on the system.

Nested Hyper-V Containers

Hyper-V for Windows Server 2022 offers nested hypervisor support. That is, the capability to run a virtual machine from within a virtual machine. This opens up many useful scenarios but also exaggerates some performance impact that the hypervisor incurs, as there are two level of hypervisors running above the physical host.

For containers, this has an impact when running a Hyper-V container inside a virtual machine. Since a Hyper-V Container offers isolation through a hypervisor layer between itself and the container host, when the container host is a Hyper-V based virtual machine, there is performance overhead associated in terms of container start-up time, storage io, network io and throughput, and CPU.

Storage

Mounted Data Volumes

Containers offer the ability to use the container host system drive for the container scratch space. However, the container scratch space has a life span equal to that of the container. That is, when the container is stopped, the scratch space and all associated data goes away.

However, there are many scenarios in which having data persist independent of container lifetime is desired. In these cases, we support mounting data volumes from the container host into the container. For Windows Server Containers, there is negligible IO path overhead associated with mounted data volumes (near native performance). However, when mounting data volumes into Hyper-V containers, there is some IO performance degradation in that path. In addition, this impact is exaggerated when running Hyper-V containers inside of virtual machines.

Scratch Space

Both Windows Server Containers and Hyper-V containers provide a 20GB dynamic VHD for the container scratch space by default. For both container types, the container OS takes up a portion of that space, and this is true for every container started. Thus it is important to remember that every container started has some storage impact, and depending on the workload can write up to 20GB of the backing storage media. Server storage configurations should be designed with this in mind.

Networking

Windows Server Containers and Hyper-V containers offer various of networking modes to best suit the needs of differing networking configurations. Each of these options presents their own performance characteristics.

Windows Network Address Translation (WinNAT)

Each container will receive an IP address from an internal, private IP prefix (for example 172.16.0.0/12). Port forwarding / mapping from the container host to container endpoints is supported. Docker creates a NAT network by default when the dockerd first runs.

Of these three modes, the NAT configuration is the most expensive network IO path, but has the least amount of configuration needed.

Windows Server containers use a Host vNIC to attach to the virtual switch. Hyper-V Containers use a Synthetic

VM NIC (not exposed to the Utility VM) to attach to the virtual switch. When containers are communicating with the external network, packets are routed through WinNAT with address translations applied, which incurs some overhead.

Transparent

Each container endpoint is directly connected to the physical network. IP addresses from the physical network can be assigned statically or dynamically using an external DHCP server.

Transparent mode is the least expensive in terms of the network IO path, and external packets are directly passed through to the container virtual NIC giving direct access to the external network.

L2 Bridge

Each container endpoint will be in the same IP subnet as the container host. The IP addresses must be assigned statically from the same prefix as the container host. All container endpoints on the host will have the same MAC address due to Layer-2 address translation.

L2 Bridge Mode is more performant than WinNAT mode as it provides direct access to the external network, but less performant than Transparent mode as it also introduces MAC address translation.

Performance Tuning Remote Desktop Session Hosts

12/16/2022 • 12 minutes to read • Edit Online

This topic discusses how to select Remote Desktop Session Host (RD Session Host) hardware, tune the host, and tune applications.

In this topic:

Selecting the proper hardware for performance

For an RD Session Host server deployment, the choice of hardware is governed by the application set and how users use them. The key factors that affect the number of users and their experience are CPU, memory, disk, and graphics. This section contains additional guidelines that are specific to RD Session Host servers and is mostly related to the multi-user environment of RD Session Host servers.

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 logical processors can help reduce abnormal CPU congestion situations, which are usually caused by a few overactive threads that are contained by a similar number of logical processors.

Therefore, the more logical processors on a system, the lower the cushion margin that must be built in to 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.

Memory configuration

Memory configuration is dependent on 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 code or data pages have little effect because only one copy is present on the system.

One interesting observation (assuming the disk system that is backing up the page file does not change) is that 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. These faults eventually overwhelm 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 configuration

Storage is one of the most overlooked aspects when you configure RD Session Host servers, and it can be the most common limitation in systems that are deployed in the field.

The disk activity that is generated on a typical RD Session Host server affects the following areas:

- System files and application binaries
- Page files
- User profiles and user data

Ideally, these areas should be backed up by distinct storage devices. Using striped RAID configurations or other types of high-performance storage further improves performance. We highly recommend that you use storage adapters with battery-backed write caching. Controllers with disk write caching offer improved support for synchronous write operations. Because all users have a separate hive, synchronous write operations are significantly more common on an RD Session Host server. 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 **Turn off Windows write-cache buffer flushing** on the device check boxes.

Network configuration

Network usage for an RD Session Host server includes two main categories:

- RD Session Host connection traffic usage is determined almost exclusively by the drawing patterns that are exhibited by the applications 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 kilobits per second, whereas rich graphics and video playback cause significant increases in bandwidth usage.

- 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 Remote Desktop Session Host

Most of the CPU usage on an RD Session Host server is driven by apps. Desktop apps 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 usage that is needed to complete an action to avoid adversely affecting other sessions.

Consider the following suggestions when you configure apps that are to be used on an RD Session Host server:

- Minimize background idle loop processing

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

- Minimize how often an app performs a state check or update.

  Disabling such behaviors or increasing the interval between polling iterations and timer firing significantly benefits CPU usage because the 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 apps by reducing their synchronization frequency.

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

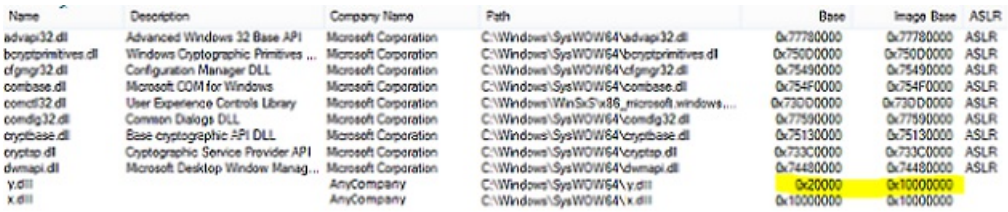
- Disable unnecessary processes that are registered to start with user sign-in or a session startup.

  These processes can significantly contribute to the cost of CPU usage when creating a new user session,

which generally is a CPU-intensive process, and it can be very expensive in morning scenarios. Use MsConfig.exe or MsInfo32.exe to obtain a list of processes that are started at user sign-in. For more detailed info, you can use Autoruns for Windows.

For memory consumption, you should consider the following:

- Verify that DLLs loaded by an app are not relocated.
  - Relocated DLLs can be verified by selecting Process DLL view, as shown in the following figure, by using Process Explorer.
  - Here we can see that y.dll was relocated because x.dll already occupied its default base address and ASLR was not enabled



Name	Description	Company Name	Path	Base	Image Base	ASLR
advapi32.dll	Advanced Windows 32 Base API	Microsoft Corporation	C:\Windows\SysWOW64\advapi32.dll	0x77780000	0x77780000	ASLR
bcryptprimitives.dll	Windows Cryptographic Primitives ...	Microsoft Corporation	C:\Windows\SysWOW64\bcryptprimitives.dll	0x750D0000	0x750D0000	ASLR
cfgmgr32.dll	Configuration Manager DLL	Microsoft Corporation	C:\Windows\SysWOW64\cfgmgr32.dll	0x75490000	0x75490000	ASLR
combase.dll	Microsoft COM for Windows	Microsoft Corporation	C:\Windows\SysWOW64\combase.dll	0x754F0000	0x754F0000	ASLR
comctl32.dll	User Experience Controls Library	Microsoft Corporation	C:\Windows\WinSxS\x86_microsoft.windows....	0x73DD0000	0x73DD0000	ASLR
comdlg32.dll	Common Dialogs DLL	Microsoft Corporation	C:\Windows\SysWOW64\comdlg32.dll	0x77590000	0x77590000	ASLR
cryptbase.dll	Base cryptographic API DLL	Microsoft Corporation	C:\Windows\SysWOW64\cryptbase.dll	0x75130000	0x75130000	ASLR
cryptsp.dll	Cryptographic Service Provider API	Microsoft Corporation	C:\Windows\SysWOW64\cryptsp.dll	0x733C0000	0x733C0000	ASLR
dwmapi.dll	Microsoft Desktop Window Manag...	Microsoft Corporation	C:\Windows\SysWOW64\dwmapi.dll	0x74480000	0x74480000	ASLR
y.dll		AnyCompany	C:\Windows\SysWOW64\y.dll	0x20000	0x10000000	
x.dll		AnyCompany	C:\Windows\SysWOW64\x.dll	0x10000000	0x10000000	

    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 issues on an RD Session Host 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.

Remote Desktop Session Host tuning parameters

Page file

Insufficient page file size can cause memory allocation failures in apps or system components. You can use the memory-to-committed bytes performance counter to monitor how much committed virtual memory is on the system.

Antivirus

Installing antivirus software on an RD Session Host server greatly affects overall system performance, especially 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.

Task Scheduler

Task Scheduler lets you examine the list of tasks that are scheduled for different events. For an RD Session Host server, it is useful to focus specifically on the tasks that are configured to run on idle, at user sign-in, or on session connect and disconnect. Because of the specifics 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 should disable any notifications by removing the component that registers them from the startup list or by changing the configuration on apps and system components to disable them. You can use **Customize Notifications Icons** to examine the list of notifications that are available on the server.

Remote Desktop Protocol data compression

Remote Desktop Protocol compression can be configured by using Group Policy under **Computer Configuration** > **Administrative Templates** > **Windows Components** > **Remote Desktop Services** >

Remote Desktop Session Host > **Remote Session Environment** > **Configure compression for RemoteFX data**. Three values are possible:

- **Optimized to use less memory** Consumes the least amount of memory per session but has the lowest compression ratio and therefore the highest bandwidth consumption.
- **Balances memory and network bandwidth** 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. If you want to use this setting, you should assess the maximum number of sessions and test to that level with this setting before you place the server in production.

You can also choose to not use a Remote Desktop Protocol compression algorithm, so we only recommend using it with a hardware device designed to optimize network traffic. Even if you choose not to use a compression algorithm, some graphics data will be compressed.

Device redirection

Device redirection can be configured by using Group Policy under **Computer Configuration** > **Administrative Templates** > **Windows Components** > **Remote Desktop Services** > **Remote Desktop Session Host** > **Device and Resource Redirection** or by using the **Session Collection** properties box in Server Manager.

Generally, device redirection increases how much network bandwidth RD Session Host server connections use because data is exchanged between devices on the client computers and processes that are running in the server session. The extent of the increase is a function of the 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 increases CPU usage at sign-in. 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.
- Introduced in Windows Server 2008, Easy Print printer driver redirection 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, Easy Print causes slightly increased network bandwidth usage, but not significant enough to offset the other performance, manageability, and reliability benefits.

Audio redirection causes a steady stream of network traffic. Audio redirection also enables users to run multimedia apps that typically have high CPU consumption.

Client experience settings

By default, Remote Desktop Connection (RDC) automatically chooses the right experience setting based on the suitability of the network connection between the server and client computers. We recommend that the RDC configuration remain at **Detect connection quality automatically**.

For advanced users, RDC provides control over a range of settings that influence network bandwidth performance for the Remote Desktop Services connection. You can access the following settings by using the **Experience** tab in Remote Desktop Connection or as settings in the RDP file.

The following settings apply when connecting to any computer:

- **Disable wallpaper** (Disable wallpaper:i:0) Does not show desktop wallpaper on redirected connections. This setting can significantly reduce bandwidth usage if desktop wallpaper consists of an image or other

content with significant costs for drawing.

- **Bitmap cache** (Bitmapcachepersistenable:i:1) When this setting is enabled, it creates a client-side cache of bitmaps that are rendered in the session. It provides a significant improvement on bandwidth usage, and it should always be enabled (unless there are other security considerations).
- **Show contents of windows while dragging** (Disable full window drag:i:1) When this setting is disabled, it reduces bandwidth by displaying only the window frame instead of all the content when the window is dragged.
- **Menu and window animation** (Disable menu anims:i:1 and Disable cursor setting:i:1): When these settings are disabled, it reduces bandwidth by disabling animation on menus (such as fading) and cursors.
- **Font smoothing** (Allow font smoothing:i:0) Controls ClearType font-rendering support. When connecting to computers running Windows 8 or Windows Server 2012 and above, enabling or disabling this setting does not have a significant impact on bandwidth usage. However, for computers running versions earlier than Windows 7 and Windows 2008 R2, enabling this setting affects network bandwidth consumption significantly.

The following settings only apply when connecting to computers running Windows 7 and earlier operating system versions:

- **Desktop composition** This setting is supported only for a remote session to a computer running Windows 7 or Windows Server 2008 R2.
- **Visual styles** (disable themes:i:1) When this setting is disabled, it reduces bandwidth by simplifying theme drawings that use the Classic theme.

By using the **Experience** tab within Remote Desktop Connection, you can choose your connection speed to influence network bandwidth performance. The following lists the options that are available to configure your connection speed:

- **Detect connection quality automatically** (Connection type:i:7) When this setting is enabled, Remote Desktop Connection automatically chooses settings that will result in optimal user experience based on connection quality. (This configuration is recommended when connecting to computers running Windows 8 or Windows Server 2012 and above).
- **Modem (56 Kbps)** (Connection type:i:1) This setting enables persistent bitmap caching.
- **Low Speed Broadband (256 Kbps - 2 Mbps)** (Connection type:i:2) This setting enables persistent bitmap caching and visual styles.
- **Cellular/Satellite (2Mbps - 16 Mbps with high latency)** (Connection type:i:3) This setting enables desktop composition, persistent bitmap caching, visual styles, and desktop background.
- **High-speed broadband (2 Mbps – 10 Mbps)** (Connection type:i:4) This setting enables desktop composition, show contents of windows while dragging, menu and window animation, persistent bitmap caching, visual styles, and desktop background.
- **WAN (10 Mbps or higher with high latency)** (Connection type:i:5) This setting enables desktop composition, show contents of windows while dragging, menu and window animation, persistent bitmap caching, visual styles, and desktop background.
- **LAN (10 Mbps or higher)** (Connection type:i:6) This setting enables desktop composition, show contents of windows while dragging, menu and window animation, persistent bitmap caching, themes, and desktop background.

Desktop Size

Desktop size for remote sessions can be controlled by using the Display tab in Remote Desktop Connection or by using the RDP configuration 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 is 4096 x 2048.

Performance Tuning Remote Desktop Session Hosts

12/16/2022 • 12 minutes to read • Edit Online

This topic discusses how to select Remote Desktop Session Host (RD Session Host) hardware, tune the host, and tune applications.

In this topic:

Selecting the proper hardware for performance

For an RD Session Host server deployment, the choice of hardware is governed by the application set and how users use them. The key factors that affect the number of users and their experience are CPU, memory, disk, and graphics. This section contains additional guidelines that are specific to RD Session Host servers and is mostly related to the multi-user environment of RD Session Host servers.

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 logical processors can help reduce abnormal CPU congestion situations, which are usually caused by a few overactive threads that are contained by a similar number of logical processors.

Therefore, the more logical processors on a system, the lower the cushion margin that must be built in to 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.

Memory configuration

Memory configuration is dependent on 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 code or data pages have little effect because only one copy is present on the system.

One interesting observation (assuming the disk system that is backing up the page file does not change) is that 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. These faults eventually overwhelm 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 configuration

Storage is one of the most overlooked aspects when you configure RD Session Host servers, and it can be the most common limitation in systems that are deployed in the field.

The disk activity that is generated on a typical RD Session Host server affects the following areas:

- System files and application binaries
- Page files
- User profiles and user data

Ideally, these areas should be backed up by distinct storage devices. Using striped RAID configurations or other types of high-performance storage further improves performance. We highly recommend that you use storage adapters with battery-backed write caching. Controllers with disk write caching offer improved support for synchronous write operations. Because all users have a separate hive, synchronous write operations are significantly more common on an RD Session Host server. 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 **Turn off Windows write-cache buffer flushing** on the device check boxes.

Network configuration

Network usage for an RD Session Host server includes two main categories:

- RD Session Host connection traffic usage is determined almost exclusively by the drawing patterns that are exhibited by the applications 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 kilobits per second, whereas rich graphics and video playback cause significant increases in bandwidth usage.

- 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 Remote Desktop Session Host

Most of the CPU usage on an RD Session Host server is driven by apps. Desktop apps 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 usage that is needed to complete an action to avoid adversely affecting other sessions.

Consider the following suggestions when you configure apps that are to be used on an RD Session Host server:

- Minimize background idle loop processing

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

- Minimize how often an app performs a state check or update.

  Disabling such behaviors or increasing the interval between polling iterations and timer firing significantly benefits CPU usage because the 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 apps by reducing their synchronization frequency.

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

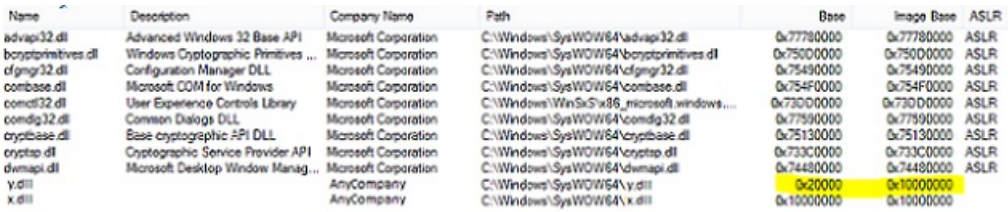
- Disable unnecessary processes that are registered to start with user sign-in or a session startup.

  These processes can significantly contribute to the cost of CPU usage when creating a new user session,

which generally is a CPU-intensive process, and it can be very expensive in morning scenarios. Use MsConfig.exe or MsInfo32.exe to obtain a list of processes that are started at user sign-in. For more detailed info, you can use Autoruns for Windows.

For memory consumption, you should consider the following:

- Verify that DLLs loaded by an app are not relocated.
  - Relocated DLLs can be verified by selecting Process DLL view, as shown in the following figure, by using Process Explorer.
  - Here we can see that y.dll was relocated because x.dll already occupied its default base address and ASLR was not enabled



Name	Description	Company Name	Path	Base	Image Base	ASLR
advapi32.dll	Advanced Windows 32 Base API	Microsoft Corporation	C:\Windows\SysWOW64\advapi32.dll	0x77780000	0x77780000	ASLR
bcryptprimitives.dll	Windows Cryptographic Primitives ...	Microsoft Corporation	C:\Windows\SysWOW64\bcryptprimitives.dll	0x750D0000	0x750D0000	ASLR
cfgmgr32.dll	Configuration Manager DLL	Microsoft Corporation	C:\Windows\SysWOW64\cfgmgr32.dll	0x75490000	0x75490000	ASLR
combase.dll	Microsoft COM for Windows	Microsoft Corporation	C:\Windows\SysWOW64\combase.dll	0x754F0000	0x754F0000	ASLR
comctl32.dll	User Experience Controls Library	Microsoft Corporation	C:\Windows\WinSxS\x86_microsoft.windows....	0x73DD0000	0x73DD0000	ASLR
comdlg32.dll	Common Dialogs DLL	Microsoft Corporation	C:\Windows\SysWOW64\comdlg32.dll	0x77590000	0x77590000	ASLR
cryptbase.dll	Base cryptographic API DLL	Microsoft Corporation	C:\Windows\SysWOW64\cryptbase.dll	0x75130000	0x75130000	ASLR
cryptsp.dll	Cryptographic Service Provider API	Microsoft Corporation	C:\Windows\SysWOW64\cryptsp.dll	0x733C0000	0x733C0000	ASLR
dwmapi.dll	Microsoft Desktop Window Manag...	Microsoft Corporation	C:\Windows\SysWOW64\dwmapi.dll	0x74480000	0x74480000	ASLR
y.dll		AnyCompany	C:\Windows\SysWOW64\y.dll	0x20000	0x10000000	
x.dll		AnyCompany	C:\Windows\SysWOW64\x.dll	0x10000000	0x10000000	

    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 issues on an RD Session Host 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.

Remote Desktop Session Host tuning parameters

Page file

Insufficient page file size can cause memory allocation failures in apps or system components. You can use the memory-to-committed bytes performance counter to monitor how much committed virtual memory is on the system.

Antivirus

Installing antivirus software on an RD Session Host server greatly affects overall system performance, especially 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.

Task Scheduler

Task Scheduler lets you examine the list of tasks that are scheduled for different events. For an RD Session Host server, it is useful to focus specifically on the tasks that are configured to run on idle, at user sign-in, or on session connect and disconnect. Because of the specifics 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 should disable any notifications by removing the component that registers them from the startup list or by changing the configuration on apps and system components to disable them. You can use **Customize Notifications Icons** to examine the list of notifications that are available on the server.

Remote Desktop Protocol data compression

Remote Desktop Protocol compression can be configured by using Group Policy under **Computer Configuration** > **Administrative Templates** > **Windows Components** > **Remote Desktop Services** >

Remote Desktop Session Host > **Remote Session Environment** > **Configure compression for RemoteFX data**. Three values are possible:

- **Optimized to use less memory** Consumes the least amount of memory per session but has the lowest compression ratio and therefore the highest bandwidth consumption.
- **Balances memory and network bandwidth** 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. If you want to use this setting, you should assess the maximum number of sessions and test to that level with this setting before you place the server in production.

You can also choose to not use a Remote Desktop Protocol compression algorithm, so we only recommend using it with a hardware device designed to optimize network traffic. Even if you choose not to use a compression algorithm, some graphics data will be compressed.

Device redirection

Device redirection can be configured by using Group Policy under **Computer Configuration** > **Administrative Templates** > **Windows Components** > **Remote Desktop Services** > **Remote Desktop Session Host** > **Device and Resource Redirection** or by using the **Session Collection** properties box in Server Manager.

Generally, device redirection increases how much network bandwidth RD Session Host server connections use because data is exchanged between devices on the client computers and processes that are running in the server session. The extent of the increase is a function of the 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 increases CPU usage at sign-in. 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.
- Introduced in Windows Server 2008, Easy Print printer driver redirection 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, Easy Print causes slightly increased network bandwidth usage, but not significant enough to offset the other performance, manageability, and reliability benefits.

Audio redirection causes a steady stream of network traffic. Audio redirection also enables users to run multimedia apps that typically have high CPU consumption.

Client experience settings

By default, Remote Desktop Connection (RDC) automatically chooses the right experience setting based on the suitability of the network connection between the server and client computers. We recommend that the RDC configuration remain at **Detect connection quality automatically**.

For advanced users, RDC provides control over a range of settings that influence network bandwidth performance for the Remote Desktop Services connection. You can access the following settings by using the **Experience** tab in Remote Desktop Connection or as settings in the RDP file.

The following settings apply when connecting to any computer:

- **Disable wallpaper** (Disable wallpaper:i:0) Does not show desktop wallpaper on redirected connections. This setting can significantly reduce bandwidth usage if desktop wallpaper consists of an image or other

content with significant costs for drawing.

- **Bitmap cache** (Bitmapcachepersistenable:i:1) When this setting is enabled, it creates a client-side cache of bitmaps that are rendered in the session. It provides a significant improvement on bandwidth usage, and it should always be enabled (unless there are other security considerations).
- **Show contents of windows while dragging** (Disable full window drag:i:1) When this setting is disabled, it reduces bandwidth by displaying only the window frame instead of all the content when the window is dragged.
- **Menu and window animation** (Disable menu anims:i:1 and Disable cursor setting:i:1): When these settings are disabled, it reduces bandwidth by disabling animation on menus (such as fading) and cursors.
- **Font smoothing** (Allow font smoothing:i:0) Controls ClearType font-rendering support. When connecting to computers running Windows 8 or Windows Server 2012 and above, enabling or disabling this setting does not have a significant impact on bandwidth usage. However, for computers running versions earlier than Windows 7 and Windows 2008 R2, enabling this setting affects network bandwidth consumption significantly.

The following settings only apply when connecting to computers running Windows 7 and earlier operating system versions:

- **Desktop composition** This setting is supported only for a remote session to a computer running Windows 7 or Windows Server 2008 R2.
- **Visual styles** (disable themes:i:1) When this setting is disabled, it reduces bandwidth by simplifying theme drawings that use the Classic theme.

By using the **Experience** tab within Remote Desktop Connection, you can choose your connection speed to influence network bandwidth performance. The following lists the options that are available to configure your connection speed:

- **Detect connection quality automatically** (Connection type:i:7) When this setting is enabled, Remote Desktop Connection automatically chooses settings that will result in optimal user experience based on connection quality. (This configuration is recommended when connecting to computers running Windows 8 or Windows Server 2012 and above).
- **Modem (56 Kbps)** (Connection type:i:1) This setting enables persistent bitmap caching.
- **Low Speed Broadband (256 Kbps - 2 Mbps)** (Connection type:i:2) This setting enables persistent bitmap caching and visual styles.
- **Cellular/Satellite (2Mbps - 16 Mbps with high latency)** (Connection type:i:3) This setting enables desktop composition, persistent bitmap caching, visual styles, and desktop background.
- **High-speed broadband (2 Mbps – 10 Mbps)** (Connection type:i:4) This setting enables desktop composition, show contents of windows while dragging, menu and window animation, persistent bitmap caching, visual styles, and desktop background.
- **WAN (10 Mbps or higher with high latency)** (Connection type:i:5) This setting enables desktop composition, show contents of windows while dragging, menu and window animation, persistent bitmap caching, visual styles, and desktop background.
- **LAN (10 Mbps or higher)** (Connection type:i:6) This setting enables desktop composition, show contents of windows while dragging, menu and window animation, persistent bitmap caching, themes, and desktop background.

Desktop Size

Desktop size for remote sessions can be controlled by using the Display tab in Remote Desktop Connection or by using the RDP configuration 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 is 4096 x 2048.

Performance Tuning Remote Desktop Virtualization Hosts

12/16/2022 • 5 minutes to read • Edit Online

Remote Desktop Virtualization Host (RD Virtualization Host) is a role service that supports Virtual Desktop Infrastructure (VDI) scenarios and lets multiple users run Windows-based applications in virtual machines hosted on a server running Windows Server and Hyper-V.

Windows Server supports two types of virtual desktops: personal virtual desktops and pooled virtual desktops.

General considerations

Storage

Storage is the most likely performance bottleneck, and it is important to size your storage to properly handle the I/O load that is generated by virtual machine state changes. If a pilot or simulation is not feasible, a good guideline is to provision one disk spindle for four active virtual machines. Use disk configurations that have good write performance (such as RAID 1+0).

When appropriate, use Disk Deduplication and caching to reduce the disk read load and to enable your storage solution to speed up performance by caching a significant portion of the image.

Data Deduplication and VDI

Introduced in Windows Server 2012 R2, Data Deduplication supports optimization of open files. In order to use virtual machines running on a deduplicated volume, the virtual machine files need to be stored on a separate host from the Hyper-V host. If Hyper-V and deduplication are running on the same machine, the two features will contend for system resources and negatively impact overall performance.

The volume must also be configured to use the "Virtual Desktop Infrastructure (VDI)" deduplication optimization type. You can configure this by using Server Manager (**File and Storage Services** -> **Volumes** -> **Dedup Settings**) or by using the following Windows PowerShell command:

```
Enable-DedupVolume <volume> -UsageType HyperV
```

> **NOTE**
> Data Deduplication optimization of open files is supported only for VDI scenarios with Hyper-V using remote storage over SMB 3.0.

Memory

Server memory usage is driven by three main factors:

- Operating system overhead
- Hyper-V service overhead per virtual machine
- Memory allocated to each virtual machine

For a typical knowledge worker workload, guest virtual machines running x86 Window 8 or Windows 8.1 should be given ~512 MB of memory as the baseline. However, Dynamic Memory will likely increase the guest virtual machine's memory to about 800 MB, depending on the workload. For x64, we see about 800 MB starting,

increasing to 1024 MB.

Therefore, it is important to provide enough server memory to satisfy the memory that is required by the expected number of guest virtual machines, plus allow a sufficient amount of memory for the server.

CPU

When you plan server capacity for an RD Virtualization Host server, the number of virtual machines per physical core will depend on the nature of the workload. As a starting point, it is reasonable to plan 12 virtual machines per physical core, and then run the appropriate scenarios to validate performance and density. Higher density may be achievable depending on the specifics of the workload.

We recommend enabling hyper-threading, but be sure to calculate the oversubscription ratio based on the number of physical cores and not the number of logical processors. This ensures the expected level of performance on a per CPU basis.

Performance optimizations

Dynamic Memory

Dynamic Memory enables more efficiently utilization of the memory resources of the server running Hyper-V by balancing how memory is distributed between running virtual machines. Memory can be dynamically reallocated between virtual machines in response to their changing workloads.

Dynamic Memory enables you to increase virtual machine density with the resources you already have without sacrificing performance or scalability. The result is more efficient use of expensive server hardware resources, which can translate into easier management and lower costs.

On guest operating systems running Windows 8 and above with virtual processors that span multiple logical processors, consider the tradeoff between running with Dynamic Memory to help minimize memory usage and disabling Dynamic Memory to improve the performance of an application that is computer-topology aware. Such an application can leverage the topology information to make scheduling and memory allocation decisions.

Tiered Storage

RD Virtualization Host supports tiered storage for virtual desktop pools. The physical computer that is shared by all pooled virtual desktops within a collection can use a small-size, high-performance storage solution, such as a mirrored solid-state drive (SSD). The pooled virtual desktops can be placed on less expensive, traditional storage such as RAID 1+0.

The physical computer should be placed on a SSD is because most of the read-I/Os from pooled virtual desktops go to the management operating system. Therefore, the storage that is used by the physical computer must sustain much higher read I/Os per second.

This deployment configuration assures cost effective performance where performance is needed. The SSD provides higher performance on a smaller size disk (~20 GB per collection, depending on the configuration). Traditional storage for pooled virtual desktops (RAID 1+0) uses about 3 GB per virtual machine.

CSV cache

Failover Clustering in Windows Server 2012 and above provides caching on Cluster Shared Volumes (CSV). This is extremely beneficial for pooled virtual desktop collections where the majority of the read I/Os come from the management operating system. The CSV cache provides higher performance by several orders of magnitude because it caches blocks that are read more than once and delivers them from system memory, which reduces the I/O. For more info on CSV cache, see How to Enable CSV Cache.

Pooled virtual desktops

By default, pooled virtual desktops are rolled back to the pristine state after a user signs out, so any changes made to the Windows operating system since the last user sign-in are abandoned.

Although it's possible to disable the rollback, it is still a temporary condition because typically a pooled virtual desktop collection is re-created due to various updates to the virtual desktop template.

It makes sense to turn off Windows features and services that depend on persistent state. Additionally, it makes sense to turn off services that are primarily for non-enterprise scenarios.

Each specific service should be evaluated appropriately prior to any broad deployment. The following are some initial things to consider:

SERVICE	WHY?
Auto update	Pooled virtual desktops are updated by re-creating the virtual desktop template.
Offline files	Virtual desktops are always online and connected from a networking point-of-view.
Background defrag	File-system changes are discarded after a user signs off (due to a rollback to the pristine state or re-creating the virtual desktop template, which results in re-creating all pooled virtual desktops).
Hibernate or sleep	No such concept for VDI
Bug check memory dump	No such concept for pooled virtual desktops. A bug-check pooled virtual desktop will start from the pristine state.
WLAN autoconfig	There is no WiFi device interface for VDI
Windows Media Player network sharing service	Consumer centric service
Home group provider	Consumer centric service
Internet connection sharing	Consumer centric service
Media Center extended services	Consumer centric service

NOTE

This list is not meant to be a complete list, because any changes will affect the intended goals and scenarios. For more info, see Hot off the presses, get it now, the Windows 8 VDI optimization script, courtesy of PFE!.

NOTE

SuperFetch in Windows 8 is enabled by default. It is VDI-aware and should not be disabled. SuperFetch can further reduce memory consumption through memory page sharing, which is beneficial for VDI. Pooled virtual desktops running Windows 7, SuperFetch should be disabled, but for personal virtual desktops running Windows 7, it should be left on.

Performance Tuning Remote Desktop Gateways

12/16/2022 • 2 minutes to read • Edit Online

> **NOTE**
> In Windows 8+ and Windows Server 2012 R2+, Remote Desktop Gateway (RD Gateway) supports TCP, UDP, and the legacy RPC transports. Most of the following data is regarding the legacy RPC transport. If the legacy RPC transport is not being used, this section is not applicable.

This topic describes the performance-related parameters that help improve the performance of a customer deployment and the tunings that rely on the customer's network usage patterns.

At its core, RD Gateway performs many packet forwarding operations between Remote Desktop Connection instances and the RD Session Host server instances within the customer's network.

> **NOTE**
> The following parameters apply to RPC transport only.

Internet Information Services (IIS) and RD Gateway export the following registry parameters to help improve system performance in the RD Gateway.

Thread tunings

- **Maxiothreads**

  ```
  HKLM\Software\Microsoft\Terminal Server Gateway\Maxiothreads (REG_DWORD)
  ```

  This app-specific thread pool specifies the number of threads that RD Gateway creates to handle incoming requests. If this registry setting is present, it takes effect. The number of threads equals the number of logical processes. If the number of logical processors is less than 5, the default is 5 threads.

- **MaxPoolThreads**

  ```
  HKLM\System\CurrentControlSet\Services\InetInfo\Parameters\MaxPoolThreads (REG_DWORD)
  ```

  This parameter specifies the number of IIS pool threads to create per logical processor. The IIS pool threads watch the network for requests and process all incoming requests. The **MaxPoolThreads** count does not include threads that RD Gateway consumes. The default value is 4.

Remote procedure call tunings for RD Gateway

The following parameters can help tune the remote procedure calls (RPC) that are received by Remote Desktop Connection and RD Gateway computers. 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\ServerReceiveWindow (REG_DWORD)
  ```

The default value is 64 KB. This value specifies the 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 a value is not present, the default value is used. When changes are made to this value, IIS must be restarted for the change to take effect.

- **ServerReceiveWindow**

  ```
  HKLM\Software\Microsoft\Rpc\ServerReceiveWindow (REG_DWORD)
  ```

  The default value is 64 KB. This value specifies the window that the client uses for data that is received from the RPC proxy. The minimum value is 8 KB, and the maximum value is 1 GB. If a value is not present, 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 RD Gateway:

- \Terminal Service Gateway*
- \RPC/HTTP Proxy*
- \RPC/HTTP Proxy Per Server*
- \Web Service*
- \W3SVC_W3WP*
- \IPv4*
- \Memory*
- \Network Interface(*)*
- \Process(*)*
- \Processor Information(*)*
- \Synchronization(*)*
- \System*
- \TCPv4*

The following performance counters are applicable only for legacy RPC transport:

- \RPC/HTTP Proxy* RPC
- \RPC/HTTP Proxy Per Server* RPC
- \Web Service* RPC
- \W3SVC_W3WP* RPC

> **NOTE**
> If applicable, add the \IPv6* and \TCPv6* objects.ReplaceThisText

Performance Tuning Web Servers

12/16/2022 • 2 minutes to read • Edit Online

This topic describes performance tuning methods and recommendations for Windows Server 2022 web servers.

Selecting the proper hardware for performance

It is important to select the proper hardware to satisfy the expected web 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 provides recommendations for hardware to avoid the following performance constraints:

- Slow CPUs offer limited processing power for CPU intensive workloads such as ASP, ASP.NET, and TLS scenarios.
- A small L2 or L3/LLC 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) can be stored, and the number of application pools or worker processes.
- Networking becomes a bottleneck because of an inefficient network adapter.
- The file system becomes a bottleneck because of an inefficient disk subsystem or storage adapter.

Operating system best practices

If possible, start with a clean installation of the operating system. Upgrading the software can 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 you must keep it, 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, place the system page file, operating system, web data, ASP template cache, and the Internet Information Services (IIS) log on separate physical disks, if possible.

To reduce contention for system resources, install Microsoft SQL Server and IIS on different servers, if possible.

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

NTFS file system settings

The system-global switch **NtfsDisableLastAccessUpdate** (REG_DWORD) 1 is located under **HKLM\System\CurrentControlSet\Control\FileSystem** and is set by default to 1. This switch reduces disk I/O load and latencies by disabling date and time stamp updating for the last file or directory access. Clean installations of Windows Server 2022, Windows Server 2019, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, and Windows Server 2008 enable this setting by default, and you do not need to adjust it. Earlier versions of Windows did not set this key. If your server is running an earlier version of Windows, or it was upgraded to Windows Server 2022, Windows Server 2019, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, or Windows

Server 2008, you should enable this setting.

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 they do not function correctly without it.

Additional References

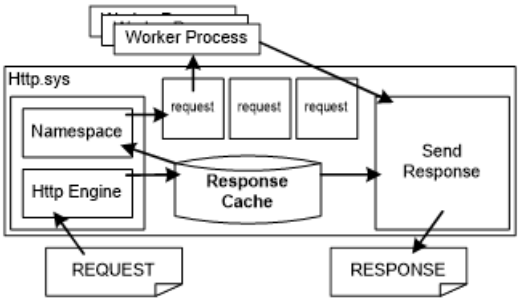
- IIS 10.0 performance tuning
- HTTP 1.1/2 tuning

Tuning IIS 10.0

12/16/2022 • 23 minutes to read • Edit Online

Internet Information Services (IIS) 10.0 is included with Windows Server 2022. It uses a process model similar to that of IIS 8.5 and IIS 7.0. A kernel-mode web driver (http.sys) receives and routes HTTP requests, and satisfies 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).

HTTP.sys is responsible for connection management and request handling. The request can be served from the HTTP.sys cache or passed to a worker process for further handling. Multiple worker processes can be configured, which provides isolation at a reduced cost. For more info on how request handling works, see the following figure:



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. Some web application platforms, such as ASP.NET, provide mechanisms to enable any dynamic content to be cached in the kernel-mode cache. The static file handler in IIS 10.0 automatically caches frequently requested files in http.sys.

Because a web server has kernel-mode and user-mode components, both components must be tuned for optimal performance. Therefore, tuning IIS 10.0 for a specific workload includes configuring the following:

- HTTP.sys and the associated kernel-mode cache
- Worker processes and user-mode IIS, including the application pool configuration
- Certain tuning parameters that affect performance

The following sections discuss how to configure the kernel-mode and user-mode aspects of IIS 10.0.

Kernel-mode settings

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 registry entry:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Http\Parameters
```

Note If the HTTP service is already running, you must restart it 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-mode cache, you can satisfy an HTTP request entirely from the kernel mode, which significantly lowers the CPU cost of handling the request. However, the kernel-mode cache of IIS 10.0 is based on physical memory, 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 consumes physical memory, whether or not the entry is being used. You must evaluate the usefulness of an item in the cache (the savings from being able to serve it from the cache) 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 non-zero value enables the kernel-mode response and fragment caching. For most workloads, the cache should remain enabled. Consider disabling the cache if you expect a very low response and fragment caching.

- **UriMaxCacheMegabyteCount** Default value: 0

  A non-zero value that specifies the maximum memory that is available to the kernel-mode cache. The default value, 0, enables the system to automatically adjust how much memory is available to the cache.

  Note Specifying the size sets only the maximum, and the system might not let the cache grow to the maximum set size.

  Â

- **UriMaxUriBytes** Default value: 262144 bytes (256 KB)

  The maximum size of an entry in the kernel-mode 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 too low causes more frequent scavenger scans, and it can result in too many flushes and cache churn.

Request and connection management settings

In Windows Server 2022, HTTP.sys manages connections automatically. The following registry settings are no longer used:

- **MaxConnections**

  ```
  HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Http\Parameters\MaxConnections
  ```

- **IdleConnectionsHighMark**

  ```
  HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Http\Parameters\IdleConnectionsHighMark
  ```

- **IdleConnectionsLowMark**

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Http\Parameters\IdleConnectionsLowMark
```

- **IdleListTrimmerPeriod**

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Http\Parameters\IdleListTrimmerPeriod
```

- **RequestBufferLookasideDepth**

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Http\Parameters\RequestBufferLookasideDepth
```

- **InternalRequestLookasideDepth**

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Http\Parameters\InternalRequestLookasideDepth
```

User-mode settings

The settings in this section affect the IISÂ 10.0 worker process behavior. Most of these settings can be found in the following XML configuration file:

%SystemRoot%\system32\inetsrv\config\applicationHost.config

Use Appcmd.exe, the IIS 10.0 Management Console, the WebAdministration or IISAdministration PowerShell Cmdlets to change them. Most settings are automatically detected, and they do not require a restart of the IIS 10.0 worker processes or web application server. For more info about the applicationHost.config file, see Introduction to ApplicationHost.config.

Ideal CPU setting for NUMA hardware

Starting from Windows Server 2016, IIS 10.0 supports automatic ideal CPU assignment for its thread pool threads to enhance the performance and scalability on NUMA hardware. This feature is enabled by default and can be configured through the following registry key:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\InetInfo\Parameters\ThreadPoolUseIdealCpu
```

With this feature enabled, IIS thread manager makes its best effort to evenly distribute IIS thread pool threads across all CPUs in all NUMA nodes based on their current loads. In general, it is recommended to keep this default setting unchanged for NUMA hardware.

Note The ideal CPU setting is different from the worker process NUMA node assignment settings (numaNodeAssignment and numaNodeAffinityMode) introduced in CPU Settings for an Application Pool. The ideal CPU setting affects how IIS distributes its thread pool threads, while the worker process NUMA node assignment settings determine on which NUMA node a worker process starts.

User-mode cache behavior settings

This section describes the settings that affect caching behavior in IISÂ 10.0. The user-mode cache is implemented as a module that listens to the global caching events that are raised by the integrated pipeline. 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 set of frequently accessed files versus the amount of RAM or the IIS process address space.	0
maxResponseSize	Caches files up to the specified size. The actual value depends on the number and size of the largest files in the data set versus the available RAM. Caching large, frequently requested files can reduce CPU usage, disk access, and associated latencies.	262144

Compression behavior settings

IIS starting from 7.0 compresses static content by default. Also, compression of dynamic content is enabled by default when the DynamicCompressionModule is installed. Compression reduces bandwidth usage but increases CPU usage. Compressed content is cached in the kernel-mode cache if possible. Starting from 8.5, IIS 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 the system.webServer/globalModules section in applicationHost.config.

system.webServer/httpCompression

ATTRIBUTE	DESCRIPTION	DEFAULT
staticCompression-EnableCpuUsage staticCompression-DisableCpuUsage dynamicCompression-EnableCpuUsage dynamicCompression-DisableCpuUsage	Enables or disables compression if the current percentage CPU usage goes above or below specified limits. Starting with IIS 7.0, compression is automatically disabled if steady-state CPU increases above the disable threshold. Compression is enabled if CPU drops below the enable threshold.	50, 100, 50, and 90 respectively

ATTRIBUTE	DESCRIPTION	DEFAULT
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.	%SystemDrive%\inetpub\temp\IIS Temporary Compressed Files
doDiskSpaceLimiting	Specifies whether a limit exists for how much disk space all compressed files can occupy. Compressed files are stored in the compression directory that is specified by the **directory** attribute.	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.	100 MB

system.webServer/urlCompression

ATTRIBUTE	DESCRIPTION	DEFAULT
doStaticCompression	Specifies whether static content is compressed.	True
doDynamicCompression	Specifies whether dynamic content is compressed.	True

> **NOTE**
> For servers running IIS 10.0 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, aroundÂ 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 searches the list of allowed default documents for each name in the file system. This can adversely affect performance, especially if reaching the content requires making a network round trip or touching a disk.

You can avoid the overhead by selectively disabling default documents and by reducing or ordering the list of documents. For websites 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.

You can selectively set the default document behavior on particular URLs by customizing the 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 sets 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.

Central binary logging

When the server session has numerous URL groups under it, the process of creating hundreds of formatted log files for individual URL groups and writing the log data to a disk can quickly consume valuable CPU and memory resources, thereby creating performance and scalability issues. Centralized binary logging minimizes the amount of system resources that are used for logging, while at the same time providing detailed log data for organizations that require it. 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 drive 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.	%SystemDrive%\inetpub\logs\LogFiles

Application and site tunings

The following settings relate to application pool and site tunings.

system.applicationHost/applicationPools/applicationPoolDefaults

ATTRIBUTE	DESCRIPTION	DEFAULT

ATTRIBUTE	DESCRIPTION	DEFAULT
queueLength	Indicates to HTTP.sys how many requests are queued for an application pool before future requests are rejected. When the 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 computer is that user-mode address space is limited to 4 GB.	False

system.applicationHost/sites/VirtualDirectoryDefault

ATTRIBUTE	DESCRIPTION	DEFAULT
allowSubDirConfig	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). By imposing a simple limitation, which allows configuration only in virtual directories, IISÂ 10.0 can 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 of websites that have a very large set of randomly accessed static content.	True

Managing IIS 10.0 modules

IIS 10.0 has been factored into multiple, user-extensible modules to support a modular structure. This factorization has a small cost. For each module the integrated pipeline must call 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 are not relevant to a particular website.

A web server that is tuned 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 sections in addition to removing the module

declaration in system.webServer/globalModules.

Classic ASP settings

The major cost of processing a classic ASP request includes initializing a script engine, compiling the requested ASP script into an ASP template, and executing the template on the script engine. While the template execution cost depends on the complexity of the requested ASP script, IIS classic ASP module can cache script engines in memory and cache templates in both memory and disk (only if in-memory template cache overflows) to boost performance in CPU-bound scenarios.

The following settings are used to configure the classic ASP template cache and script engine cache, and they do not affect ASP.NET settings.

system.webServer/asp/cache

ATTRIBUTE	DESCRIPTION	DEFAULT
diskTemplateCacheDirectory	The name of the directory that ASP uses to store compiled templates when the in-memory cache overflows. Recommendation: Set to a directory that is not heavily used, for example, a drive that is not shared with the operating system, IIS log, or other frequently accessed content.	%SystemDrive%\inetpub\temp\ASP Compiled Templates
maxDiskTemplateCacheFiles	Specifies the maximum number of compiled ASP templates that can be cached on disk. Recommendation: Set to the maximum value of 0x7FFFFFFF.	2000
scriptFileCacheSize	This attribute specifies the maximum number of compiled ASP templates that can be cached in memory. Recommendation: Set to at least as many as the number of frequently-requested ASP scripts served by an application pool. If possible, set to as many ASP templates as memory limits allow.	500
scriptEngineCacheMax	Specifies the maximum number of script engines that will keep cached in memory. Recommendation: Set to at least as many as the number of frequently-requested ASP scripts served by an application pool. If possible, set to as many script engines as the memory limit allows.	250

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, which can cause errors when it is serving requests or cause under-usage of CPU resources.	25

system.webServer/asp/comPlus

ATTRIBUTE	DESCRIPTION	DEFAULT
executeInMta	Set to **True** if errors or failures are detected while IIS is serving 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 multi-threaded apartment model in ASP.	False

ASP.NET concurrency setting

ASP.NET 3.5

By default, ASP.NET limits request concurrency to reduce steady-state memory consumption on the server. High concurrency applications might need to adjust some settings to improve overall performance. You can change this setting in aspnet.config file:

```
<system.web>
  <applicationPool maxConcurrentRequestsPerCPU="5000"/>
</system.web>
```

The following setting is useful to fully use resources on a system:

- **maxConcurrentRequestPerCpu** Default value: 5000

  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. Consider increasing this limit on systems that run applications that perform long, synchronous I/O operations. Otherwise, users can experience high latency because of queuing or request failures due to exceeding queue limits under a high load when the default setting is used.

ASP.NET 4.6

Besides the maxConcurrentRequestPerCpu setting, ASP.NET 4.7 also provides settings to improve the performance in the applications which heavily rely on asynchronous operation. The setting can be changed in aspnet.config file.

```
<system.web>
  <applicationPool percentCpuLimit="90" percentCpuLimitMinActiveRequestPerCpu="100"/>
</system.web>
```

- **percentCpuLimit** Default value: 90 Asynchronous request has some scalability issues when a huge load

(beyond the hardware capabilities) is put on such scenario. The problem is due to the nature of allocation on asynchronous scenarios. In these conditions, allocation will happen when the asynchronous operation starts, and it will be consumed when it completes. By that time, itâs very possible the objects have been moved to generation 1 or 2 by GC. When this happens, increasing the load will show increase on request per second (rps) until a point. Once we pass that point, the time spent in GC will start to become a problem and the rps will start to dip, having a negative scaling effect. To fix the problem, when the cpu usage exceeds percentCpuLimit setting, requests will be sent to the ASP.NET native queue.

- **percentCpuLimitMinActiveRequestPerCpu** Default value: 100 CPU throttling(percentCpuLimit setting) is not based on number of requests but on how expensive they are. As a result, there could be just a few CPU-intensive requests causing a backup in the native queue with no way to empty it aside from incoming requests. To solve this problem, percentCpuLimitMinActiveRequestPerCpu can be used to ensure a minimum number of requests are being served before throttling kicks in.

Worker process and recycling options

You can configure options for recycling IIS worker processes and provide practical solutions to acute situations or events without requiring intervention or resetting a service or computer. 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 recycling can be turned off or the system can be configured to recycle very infrequently.

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. The **recycling/periodicRestart** element is per-application, which means that each attribute in the following table is partitioned on a per-application basis.

system.applicationHost/applicationPools/ApplicationPoolDefaults/recycling/periodicRestart

ATTRIBUTE	DESCRIPTION	DEFAULT
memory	Enable process recycling if virtual memory consumption exceeds the specified limit in kilobytes. This is a useful setting for 32-bit computers that have a small, 2 GB address space. It can help avoid failed requests due to out-of-memory errors.	0
privateMemory	Enable process recycling if private memory allocations exceed a specified limit in kilobytes.	0
requests	Enable process recycling after a certain number of requests.	0
time	Enable process recycling after a specified time period.	29:00:00

Dynamic worker-process page-out tuning

Starting in Windows Server 2012 R2, IIS offers the option of configuring worker process to suspend after they have been idle for a while (in addition to the option of terminate, which existed since IIS 7).

The main purpose of both the idle worker process page-out and idle worker process termination features is to conserve memory utilization on the server, since a site can consume a lot of memory even if it's just sitting there, listening. Depending on the technology used on the site (static content vs ASP.NET vs other frameworks), the memory used can be anywhere from about 10 MB to hundreds of MBs, and this means that if your server is configured with many sites, figuring out the most effective settings for your sites can dramatically improve performance of both active and suspended sites.

Before we go into specifics, we must keep in mind that if there are no memory constraints, then it's probably best to simply set the sites to never suspend or terminate. After all, thereâs little value in terminating a worker process if it's the only one on the machine.

> **NOTE**
> In case the site runs unstable code, such as code with a memory leak, or otherwise unstable, setting the site to terminate on idle can be a quick-and-dirty alternative to fixing the code bug. This isn't something we would encourage, but in a crunch, it may be better to use this feature as a clean-up mechanism while a more permanent solution is in the works.]

Another factor to consider is that if the site does use a lot of memory, then the suspension process itself takes a toll, because the computer has to write the data used by the worker process to disk. If the worker process is using a large chunk of memory, then suspending it might be more expensive than the cost of having to wait for it to start back up.

To make the best of the worker process suspension feature, you need to review your sites in each application pool, and decide which should be suspended, which should be terminated, and which should be active indefinitely. For each action and each site, you need to figure out the ideal time-out period.

Ideally, the sites that you will configure for suspension or termination are those that have visitors every day, but not enough to warrant keeping it active all the time. These are usually sites with around 20 unique visitors a day or less. You can analyze the traffic patterns using the site's log files and calculate the average daily traffic.

Keep in mind that once a specific user connects to the site, they will typically stay on it for at least a while, making additional requests, and so just counting daily requests may not accurately reflect the real traffic patterns. To get a more accurate reading, you can also use a tool, such as Microsoft Excel, to calculate the average time between requests. For example:

NUMBER	REQUEST URL	REQUEST TIME	DELTA
1	/SourceSilverLight/Geosource.web/grosource.html	10:01	
2	/SourceSilverLight/Geosource.web/sliverlight.js	10:10	0:09
3	/SourceSilverLight/Geosource.web/clientbin/geo/1.aspx	10:11	0:01
4	/IClientAccessPolicy.xml	10:12	0:01
5	/SourceSilverLight/GeosourcewebService/Service.asmx	10:23	0:11
6	/SourceSilverLight/Geosource.web/GeoSearchServer...¦.	11:50	1:27

NUMBER	REQUEST URL	REQUEST TIME	DELTA
7	/rest/Services/CachedServices/Silverlight_load_la...¦	12:50	1:00
8	/rest/Services/CachedServices/Silverlight_basemap...¦.	12:51	0:01
9	/rest/Services/DynamicService/ Silverlight_basemap...¦.	12:59	0:08
10	/rest/Services/CachedServices/Ortho_2004_cache.as...	13:40	0:41
11	/rest/Services/CachedServices/Ortho_2005_cache.js	13:40	0:00
12	/rest/Services/CachedServices/OrthoBaseEngine.aspx	13:41	0:01

The hard part, though, is figuring out what setting to apply to make sense. In our case, the site gets a bunch of requests from users, and the table above shows that a total of 4 unique sessions occurred in a period of 4 hours. With the default settings for worker process suspension of the application pool, the site would be terminated after the default timeout of 20 minutes, which means each of these users would experience the site spin-up cycle. This makes it an ideal candidate for worker process suspension, because for most of the time, the site is idle, and so suspending it would conserve resources, and allow the users to reach the site almost instantly.

A final, and very important note about this is that disk performance is crucial for this feature. Because the suspension and wake-up process involve writing and reading large amount of data to the hard drive, we strongly recommend using a fast disk for this. Solid State Drives (SSDs) are ideal and highly recommended for this, and you should make sure that the Windows page file is stored on it (if the operating system itself is not installed on the SSD, configure the operating system to move the page file to it).

Whether you use an SSD or not, we also recommend fixing the size of the page file to accommodate writing the page-out data to it without file-resizing. Page-file resizing might happen when the operating system needs to store data in the page file, because by default, Windows is configured to automatically adjust its size based on need. By setting the size to a fixed one, you can prevent resizing and improve performance a lot.

To configure a pre-fixed page file size, you need to calculate its ideal size, which depends on how many sites you will be suspending, and how much memory they consume. If the average is 200 MB for an active worker process and you have 500 sites on the servers that will be suspending, then the page file should be at least (200 * 500) MB over the base size of the page file (so base + 100 GB in our example).

> **NOTE**
>
> When sites are suspended, they will consume approximately 6 MB each, so in our case, memory usage if all sites are suspended would be around 3 GB. In reality, though, youâre probably never going to have them all suspended at the same time.

Transport Layer Security tuning parameters

The use of Transport Layer Security (TLS) imposes additional CPU cost. The most expensive component of TLS is the cost of establishing a session establishment because it involves a full handshake. Reconnection, encryption, and decryption also add to the cost. For better TLS performance, do the following:

- Enable HTTP keep-alives for TLS sessions. This eliminates the session establishment costs.
- Reuse sessions when appropriate, especially with non-keep-alive traffic.
- Selectively apply encryption only to pages or parts of the site that need it, rather to the entire site.

> **NOTE**
>
> - Larger keys provide more security, but they also use more CPU time.
> - All components might not need to be encrypted. However, mixing plain HTTP and HTTPS might result in a pop-up warning that not all content on the page is secure.

Internet Server Application Programming Interface (ISAPI)

No special tuning parameters are needed for ISAPI applications. If you write a private ISAPI extension, make sure that it is written for performance and resource use.

Managed code tuning guidelines

The integrated pipeline model in IIS 10.0 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 they 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, custom modules add significant CPU cost due to marshaling data between native and managed code. If possible, you should set preCondition to managedHandler for managed module.

To get better cold startup performance, make sure that you precompile the ASP.NET web site or leverage IIS Application Initialization feature to warm up the application.

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

If there are many I/O bound operations, try to use asynchronous version of relevant APIs which will give you much better scalability.

Also using Output Cache properly will also boost the performance of your web site.

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 has enough RAM for the expected number of concurrently running application pools. Consider using multiple application domains instead of multiple isolated processes.

Other issues that affect IIS performance

The following issues can 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. ISAPI filters that were written before IISÂ 6.0 can cause this behavior.

- Common Gateway Interface (CGI) requests

  For performance reasons, the use of CGI applications to serve requests is not recommended with IIS. Frequently creating and deleting CGI processes involves significant overhead. Better alternatives include using FastCGI, ISAPI application scripts and ASP or ASP.NET scripts. Isolation is available for each of these

options.

Additional References

- Web Server performance tuning
- HTTP 1.1/2 tuning

Performance Tuning HTTP 1.1/2

12/16/2022 • 2 minutes to read • Edit Online

HTTP/2 is meant to improve performance on the client side (e.g., page load time on a browser). On the server, it may represent a slight increase in CPU cost. Whereas the server no longer requires a single TCP connection for every request, some of that state will now be kept in the HTTP layer. Furthermore, HTTP/2 has header compression, which represents additional CPU load.

Some situations require an HTTP/1.1 fallback (resetting the HTTP/2 connection and instead establishing a new connection to use HTTP/1.1). In particular, TLS renegotiation and HTTP authentication (other than Basic and Digest) require HTTP/1.1 fallback. Even though this adds overhead, these operations already imply some delay and so are not particularly performance-sensitive.

Additional References

- Web Server performance tuning
- IIS 10.0 performance tuning

Performance Tuning Cache and Memory Manager

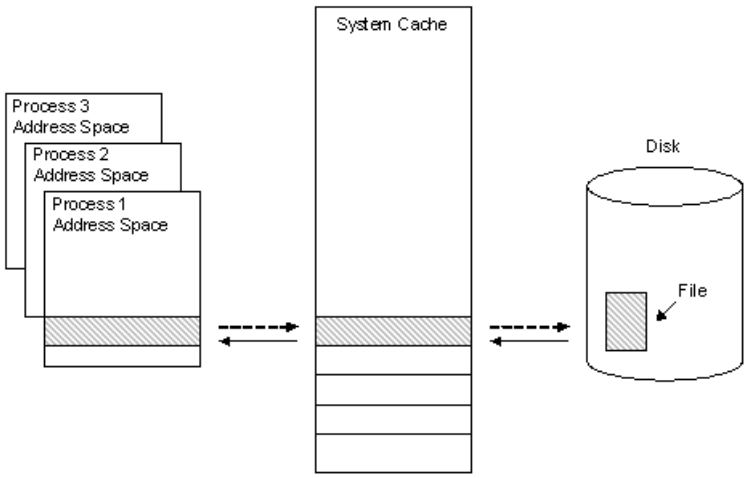
12/16/2022 • 2 minutes to read • Edit Online

By default, Windows caches file data that is read from disks and written to disks. This implies that read operations read file data from an area in system memory, known as the system file cache, rather than from the physical disk. Correspondingly, write operations write file data to the system file cache rather than to the disk, and this type of cache is referred to as a write-back cache. Caching is managed per file object. Caching occurs under the direction of the Cache Manager, which operates continuously while Windows is running.

File data in the system file cache is written to the disk at intervals determined by the operating system. Flushed pages stay either in system cache working set (when FILE_FLAG_RANDOM_ACCESS is set and file handle wasn't closed) or on the standby list where these become part of available memory.

The policy of delaying the writing of the data to the file and holding it in the cache until the cache is flushed is called lazy writing, and it is triggered by the Cache Manager at a determinate time interval. The time at which a block of file data is flushed is partially based on the amount of time it has been stored in the cache and the amount of time since the data was last accessed in a read operation. This ensures that file data that is frequently read will stay accessible in the system file cache for the maximum amount of time.

This file data caching process is illustrated in the following figure:



As depicted by the solid arrows in the preceding figure, a 256 KB region of data is read into a 256 KB cache slot in system address space when it is first requested by the Cache Manager during a file read operation. A user-mode process then copies the data in this slot to its own address space. When the process has completed its data access, it writes the altered data back to the same slot in the system cache, as shown by the dotted arrow between the process address space and the system cache. When the Cache Manager has determined that the data will no longer be needed for a certain amount of time, it writes the altered data back to the file on the disk, as shown by the dotted arrow between the system cache and the disk.

In this section:

- Cache and Memory Manager Potential Performance Issues
- Cache and Memory Manager Improvements in Windows Server 2016

Troubleshoot Cache and Memory Manager Performance Issues

12/16/2022 • 3 minutes to read • Edit Online

Before Windows Server 2012, two primary potential issues caused system file cache to grow until available memory was almost depleted under certain workloads. When this situation results in the system being sluggish, you can determine whether the server is facing one of these issues.

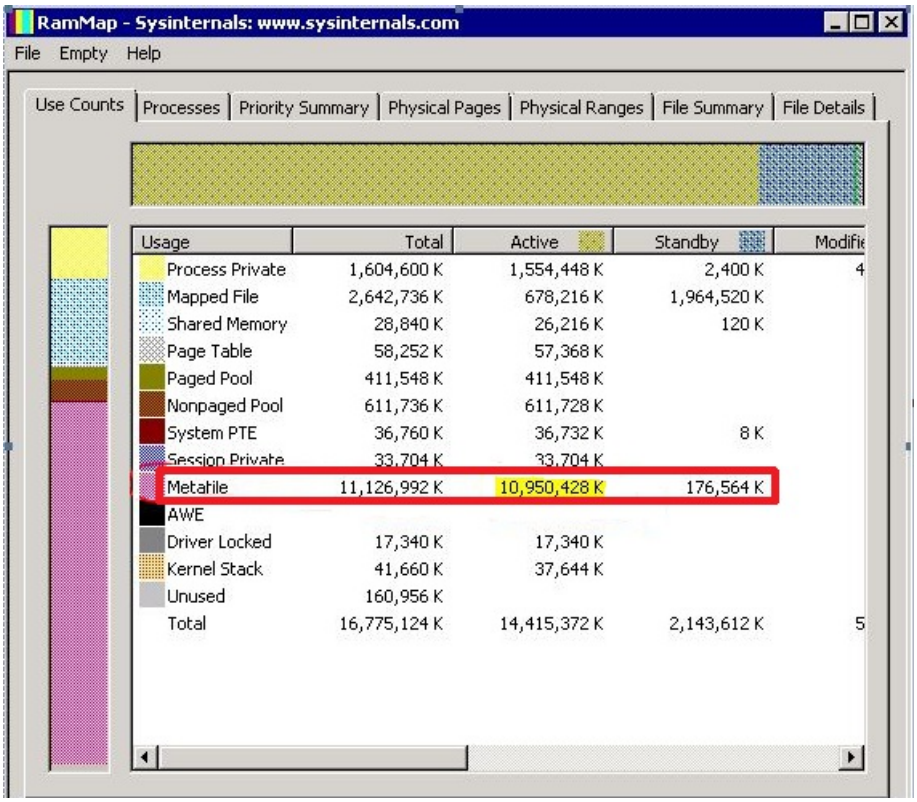
Counters to monitor

- Memory\Long-Term Average Standby Cache Lifetime (s) < 1800 seconds
- Memory\Available Mbytes is low
- Memory\System Cache Resident Bytes

If Memory\Available Mbytes is low and at the same time Memory\System Cache Resident Bytes is consuming significant part of the physical memory, you can use RAMMAP to find out what the cache is being used for.

System file cache contains NTFS metafile data structures

This problem is indicated by a high number of active Metafile pages in RAMMAP output, as shown in the following figure. This problem might have been observed on busy servers with millions of files being accessed, thereby resulting in caching NTFS metafile data not being released from the cache.



The problem used to be mitigated by *DynCache* tool. In Windows Server 2012+, the architecture has been redesigned and this problem should no longer exist.

System file cache contains memory mapped files

This problem is indicated by a high number of active Mapped file pages in RAMMAP output. This usually indicates that some application on the server is opening numerous large files using CreateFile API with FILE_FLAG_RANDOM_ACCESS flag set.

This issue is described in detail in KB article 2549369. FILE_FLAG_RANDOM_ACCESS flag is a hint for Cache Manager to keep mapped views of the file in memory as long as possible (until Memory Manager doesn't signal low memory condition). At the same time, this flag instructs Cache Manager to disable prefetching of file data.

This situation has been mitigated to some extent by working set trimming improvements in Windows Server 2012+, but the issue itself needs to be primarily addressed by the application vendor by not using FILE_FLAG_RANDOM_ACCESS. An alternative solution for the app vendor might be to use low memory priority when accessing the files. This can be achieved using the SetThreadInformation API. Pages that are accessed at low memory priority are removed from the working set more aggressively.

Cache Manager, starting in Windows Server 2016 further mitigates this by ignoring FILE_FLAG_RANDOM_ACCESS when making trimming decisions, so it is treated just like any other file opened without the FILE_FLAG_RANDOM_ACCESS flag (Cache Manager still honors this flag to disable prefetching of file data). You can still cause system cache bloat if you have large number of files opened with this flag and accessed in truly random fashion. It is highly recommended that FILE_FLAG_RANDOM_ACCESS not be used by applications.

Remote file dirty page threshold is consistently exceeded

This problem is indicated if a system experiences occasional slowdowns during writes from a remote client. This issue may occur when a large amount of data is written from a fast remote client to a slow server destination.

Prior to Windows Server 2016, in such a scenario, if the dirty page threshold in the cache is reached, further writes will behave as if there were write-through. This can cause a flush of a large amount of data to the disk, which can lead to long delays if the storage is slow, resulting in timeouts for the remote connection.

In Window Server 2016 and forward, a mitigation is put in place to reduce the likelihood of timeouts. A separate dirty page threshold for remote writes is implemented, and an inline flush will be performed when it is exceeded. This can result on occasional slowdowns during heavy write activity, but eliminates the risk of a timeout in most cases. This remote dirty page threshold is **5 GB per file** by default. For some configurations and workloads, a different number will perform better.

This threshold can be controlled with the following regkey: HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\RemoteFileDirtyPageThreshold. If the default size of 5 GB does not work well for your configuration, it is recommended to try increasing the limit in 256-MB increments until performance is satisfactory. Please note the following:

- A reboot is required for changes to this regkey to take effect.
- The units of RemoteFileDirtyPageThreshold are **number of pages** (with page size as managed by Cache Manager). This means it should be set to the desired size in bytes, divided by 4096.
- Recommended values are 128MB <= N <= 50% of available memory.
- This threshold can be disabled completely by setting it to -1. **This is not recommended** as it can result in timeouts for remote connections.

Cache and Memory Manager Improvements

12/16/2022 • 2 minutes to read • Edit Online

This topic describes Cache Manager and Memory Manager improvements in Windows Server 2012 and 2016.

Cache Manager improvements in Windows Server 2022

Cache manager is now NUMA aware, which ensures the system is better at avoiding data movement across NUMA boundaries. Avoiding accesses from a NUMA node to memory and other kernel resources on another NUMA node adds a lot of overhead. By making Cache manager NUMA aware, we have eliminated such cross-NUMA hops, thus optimizing cached IO workloads running on multi node configurations.

Cache Manager improvements in Windows Server 2019

We added Zero-Copy support for Persistent Memory (PMEM) Storage. For more information on persistent memory, see Understand and deploy persistent memory. In Direct Access (DAX) mode, PMEM operates like memory to get the lowest latency, wherein we eliminate an extra copy of data and bypass a lot of overhead from Filesystem Mini-Filters and the storage stack. This mode only works with NTFS as a filesystem.

Cache Manager improvements in Windows Server 2016

Cache Manager also added support for true Asynchronous Cached Reads. This could potentially improve the performance of an application if it relies heavily on asynchronous cached reads. While most in-box filesystems have supported async-cached reads for a while, there were often performance limitations due to various design choices related to handling of thread-pools and filesystems' internal work queues. With support from kernel-proper, Cache Manager now hides all the thread-pool and work queue management complexities from filesystems making it more efficient at handling asynchronous cached reads. Cache Manager has one set of control data structures for each of (system supported maximum) VHD-nesting levels to maximize parallelism.

Cache Manager improvements in Windows Server 2012

In addition to Cache Manager enhancements to read ahead logic for sequential workloads, a new API CcSetReadAheadGranularityEx was added to let file system drivers, such as SMB, change their read ahead parameters. It allows better throughput for remote file scenarios by sending multiple small-sized read ahead requests instead of sending a single large read ahead request. Only kernel components, such as file system drivers, can programmatically configure these values on a per-file basis.

Memory Manager improvements in Windows Server 2012

Enabling page combining may reduce memory usage on servers, which have a lot of private, pageable pages with identical contents. For example, servers running multiple instances of the same memory-intensive app, or a single app that works with highly repetitive data, might be good candidates to try page combining. The downside of enabling page combining is increased CPU usage.

Here are some examples of server roles where page combining is unlikely to give much benefit:

- File servers (most of the memory is consumed by file pages which are not private and therefore not combinable)
- Microsoft SQL Servers that are configured to use AWE or large pages (most of the memory is private but non-pageable)

Page combining is disabled by default but can be enabled by using the Enable-MMAgent Windows PowerShell cmdlet. Page combining was added in Windows Server 2012.

Network Subsystem Performance Tuning

12/16/2022 • 2 minutes to read • Edit Online

Applies to: Windows Server 2022, Windows Server 2019, Windows Server 2016, Azure Stack HCI, versions 21H2 and 20H2

You can use this topic for an overview of the network subsystem and for links to other topics in this guide.

NOTE

In addition to this topic, the following sections of this guide provide performance tuning recommendations for network devices and the network stack.

- Choosing a Network Adapter
- Configure the Order of Network Interfaces
- Performance Tuning Network Adapters
- Network-Related Performance Counters
- Performance Tools for Network Workloads

Performance tuning the network subsystem, particularly for network intensive workloads, can involve each layer of the network architecture, which is also called the network stack. These layers are broadly divided into the following sections.

1. **Network interface**. This is the lowest layer in the network stack, and contains the network driver that communicates directly with the network adapter.
2. **Network Driver Interface Specification (NDIS)**. NDIS exposes interfaces for the driver below it and for the layers above it, such as the Protocol Stack.
3. **Protocol Stack**. The protocol stack implements protocols such as TCP/IP and UDP/IP. These layers expose the transport layer interface for layers above them.
4. **System Drivers**. These are typically clients that use a transport data extension (TDX) or Winsock Kernel (WSK) interface to expose interfaces to user-mode applications. The WSK interface was introduced in Windows Server 2008 and Windows® Vista, and it is exposed by AFD.sys. The interface improves performance by eliminating the switching between user mode and kernel mode.
5. **User-Mode Applications**. These are typically Microsoft solutions or custom applications.

The table below provides a vertical illustration of the layers of the network stack, including examples of items that run in each layer.

5	**User-Mode Applications**	WMS	DNS	IIS
4	**System Drivers**	AFD.sys	HTTP.sys	
3	**Protocol Stack**	TCP/IP	UDP/IP	VPN
2	**NDIS**	Network Driver Interface Specification (NDIS)		
1	**Network interface**	Network driver		

Choosing a Network Adapter

12/16/2022 • 10 minutes to read • Edit Online

Applies to: Windows Server 2022, Windows Server 2019, Windows Server 2016, Azure Stack HCI, versions 21H2 and 20H2

You can use this topic to learn some of the features of network adapters that might affect your purchasing choices.

Network-intensive applications require high-performance network adapters. This section explores some considerations for choosing network adapters, as well as how to configure different network adapter settings to achieve the best network performance.

TIP

You can configure network adapter settings by using Windows PowerShell. For more information, see Network Adapter Cmdlets in Windows PowerShell.

Offload Capabilities

Offloading tasks from the central processing unit (CPU) to the network adapter can reduce CPU usage on the server, which improves the overall system performance.

The network stack in Microsoft products can offload one or more tasks to a network adapter if you select a network adapter that has the appropriate offload capabilities. The following table provides a brief overview of different offload capabilities that are available in Windows Server 2016.

OFFLOAD TYPE	DESCRIPTION
Checksum calculation for TCP	The network stack can offload the calculation and validation of Transmission Control Protocol (TCP) checksums on send and receive code paths. It can also offload the calculation and validation of IPv4 and IPv6 checksums on send and receive code paths.
Checksum calculation for UDP	The network stack can offload the calculation and validation of User Datagram Protocol (UDP) checksums on send and receive code paths.
Checksum calculation for IPv4	The network stack can offload the calculation and validation of IPv4 checksums on send and receive code paths.
Checksum calculation for IPv6	The network stack can offload the calculation and validation of IPv6 checksums on send and receive code paths.
Segmentation of large TCP packets	The TCP/IP transport layer supports Large Send Offload v2 (LSOv2). With LSOv2, the TCP/IP transport layer can offload the segmentation of large TCP packets to the network adapter.

OFFLOAD TYPE	DESCRIPTION
Receive Side Scaling (RSS)	RSS is a network driver technology that enables the efficient distribution of network receive processing across multiple CPUs in multiprocessor systems. More detail about RSS is provided later in this topic.
Receive Segment Coalescing (RSC)	RSC is the ability to group packets together to minimize the header processing that is necessary for the host to perform. A maximum of 64 KB of received payload can be coalesced into a single larger packet for processing. More detail about RSC is provided later in this topic.

Receive Side Scaling

Windows Server 2016, Windows Server 2012, Windows Server 2012 R2, Windows Server 2008 R2, and Windows Server 2008 support Receive Side Scaling (RSS).

Some servers are configured with multiple logical processors that share hardware resources (such as a physical core) and which are treated as Simultaneous Multi-Threading (SMT) peers. Intel Hyper-Threading Technology is an example. RSS directs network processing to up to one logical processor per core. For example, on a server with Intel Hyper-Threading, 4 cores, and 8 logical processors, RSS uses no more than 4 logical processors for network processing.

RSS distributes incoming network I/O packets among logical processors so that packets which belong to the same TCP connection are processed on the same logical processor, which preserves ordering.

RSS also load balances UDP unicast and multicast traffic, and it routes related flows (which are determined by hashing the source and destination addresses) to the same logical processor, preserving the order of related arrivals. This helps improve scalability and performance for receive-intensive scenarios for servers that have fewer network adapters than they do eligible logical processors.

Configuring RSS

In Windows Server 2016, you can configure RSS by using Windows PowerShell cmdlets and RSS profiles.

You can define RSS profiles by using the **–Profile** parameter of the **Set-NetAdapterRss** Windows PowerShell cmdlet.

Windows PowerShell commands for RSS configuration

The following cmdlets allow you to see and modify RSS parameters per network adapter.

> **NOTE**
> For a detailed command reference for each cmdlet, including syntax and parameters, you can click the following links. In addition, you can pass the cmdlet name to **Get-Help** at the Windows PowerShell prompt for details on each command.

- Disable-NetAdapterRss. This command disables RSS on the network adapter that you specify.
- Enable-NetAdapterRss. This command enables RSS on the network adapter that you specify.
- Get-NetAdapterRss. This command retrieves RSS properties of the network adapter that you specify.
- Set-NetAdapterRss. This command sets the RSS properties on the network adapter that you specify.

RSS profiles

You can use the **–Profile** parameter of the Set-NetAdapterRss cmdlet to specify which logical processors are assigned to which network adapter. Available values for this parameter are:

- **Closest**. Logical processor numbers that are near the network adapter's base RSS processor are preferred. With this profile, the operating system might rebalance logical processors dynamically based on load.
- **ClosestStatic**. Logical processor numbers near the network adapter's base RSS processor are preferred. With this profile, the operating system does not rebalance logical processors dynamically based on load.
- **NUMA**. Logical processor numbers are generally selected on different NUMA nodes to distribute the load. With this profile, the operating system might rebalance logical processors dynamically based on load.
- **NUMAStatic**. This is the **default profile**. Logical processor numbers are generally selected on different NUMA nodes to distribute the load. With this profile, the operating system will not rebalance logical processors dynamically based on load.
- **Conservative**. RSS uses as few processors as possible to sustain the load. This option helps reduce the number of interrupts.

Depending on the scenario and the workload characteristics, you can also use other parameters of the **Set-NetAdapterRss** Windows PowerShell cmdlet to specify the following:

- On a per-network adapter basis, how many logical processors can be used for RSS.
- The starting offset for the range of logical processors.
- The node from which the network adapter allocates memory.

Following are the additional **Set-NetAdapterRss** parameters that you can use to configure RSS:

> **NOTE**
> In the example syntax for each parameter below, the network adapter name **Ethernet** is used as an example value for the **–Name** parameter of the **Set-NetAdapterRss** command. When you run the cmdlet, ensure that the network adapter name that you use is appropriate for your environment.

- *** MaxProcessors**: Sets the maximum number of RSS processors to be used. This ensures that application traffic is bound to a maximum number of processors on a given interface. Example syntax:

  ```
  Set-NetAdapterRss –Name "Ethernet" –MaxProcessors <value>
  ```

- *** BaseProcessorGroup**: Sets the base processor group of a NUMA node. This impacts the processor array that is used by RSS. Example syntax:

  ```
  Set-NetAdapterRss –Name "Ethernet" –BaseProcessorGroup <value>
  ```

- *** MaxProcessorGroup**: Sets the Max processor group of a NUMA node. This impacts the processor array that is used by RSS. Setting this would restrict a maximum processor group so that load balancing is aligned within a k-group. Example syntax:

  ```
  Set-NetAdapterRss –Name "Ethernet" –MaxProcessorGroup <value>
  ```

- *** BaseProcessorNumber**: Sets the base processor number of a NUMA node. This impacts the processor array that is used by RSS. This allows partitioning processors across network adapters. This is the first logical processor in the range of RSS processors that is assigned to each adapter. Example syntax:

  ```
  Set-NetAdapterRss –Name "Ethernet" –BaseProcessorNumber <Byte Value>
  ```

- *** NumaNode**: The NUMA node that each network adapter can allocate memory from. This can be within a k-group or from different k-groups. Example syntax:

```
Set-NetAdapterRss –Name "Ethernet" –NumaNodeID <value>
```

- *** NumberofReceiveQueues**: If your logical processors seem to be underutilized for receive traffic (for example, as viewed in Task Manager), you can try increasing the number of RSS queues from the default of 2 to the maximum that is supported by your network adapter. Your network adapter may have options to change the number of RSS queues as part of the driver. Example syntax:

  ```
  Set-NetAdapterRss –Name "Ethernet" –NumberOfReceiveQueues <value>
  ```

For more information, click the following link to download Scalable Networking: Eliminating the Receive Processing Bottleneck—Introducing RSS in Word format.

Understanding RSS Performance

Tuning RSS requires understanding the configuration and the load-balancing logic. To verify that the RSS settings have taken effect, you can review the output when you run the **Get-NetAdapterRss** Windows PowerShell cmdlet. Following is example output of this cmdlet.

```
PS C:\Users\Administrator> get-netadapterrss
Name                           : testnic 2
InterfaceDescription           : Broadcom BCM5708C NetXtreme II GigE (NDIS VBD Client) #66
Enabled                        : True
NumberOfReceiveQueues          : 2
Profile                        : NUMAStatic
BaseProcessor: [Group:Number]  : 0:0
MaxProcessor: [Group:Number]   : 0:15
MaxProcessors                  : 8

IndirectionTable: [Group:Number]:
     0:0    0:4    0:0    0:4    0:0    0:4    0:0    0:4
…
(# indirection table entries are a power of 2 and based on # of processors)
…
                         0:0    0:4    0:0    0:4    0:0    0:4    0:0    0:4
```

In addition to echoing parameters that were set, the key aspect of the output is the indirection table output. The indirection table displays the hash table buckets that are used to distribute incoming traffic. In this example, the n:c notation designates the Numa K-Group:CPU index pair that is used to direct incoming traffic. We see exactly 2 unique entries (0:0 and 0:4), which represent k-group 0/cpu0 and k-group 0/cpu 4, respectively.

There is only one k-group for this system (k-group 0) and a n (where n <= 128) indirection table entry. Because the number of receive queues is set to 2, only 2 processors (0:0, 0:4) are chosen - even though maximum processors is set to 8. In effect, the indirection table is hashing incoming traffic to only use 2 CPUs out of the 8 that are available.

To fully utilize the CPUs, the number of RSS Receive Queues must be equal to or greater than Max Processors. In the previous example, the Receive Queue should be set to 8 or greater.

NIC Teaming and RSS

RSS can be enabled on a network adapter that is teamed with another network interface card using NIC Teaming. In this scenario, only the underlying physical network adapter can be configured to use RSS. A user cannot set RSS cmdlets on the teamed network adapter.

Receive Segment Coalescing (RSC)

Receive Segment Coalescing (RSC) helps performance by reducing the number of IP headers that are processed for a given amount of received data. It should be used to help scale the performance of received data by grouping (or coalescing) the smaller packets into larger units.

This approach can affect latency with benefits mostly seen in throughput gains. RSC is recommended to

increase throughput for received heavy workloads. Consider deploying network adapters that support RSC.

On these network adapters, ensure that RSC is on (this is the default setting), unless you have specific workloads (for example, low latency, low throughput networking) that show benefit from RSC being off.

Understanding RSC Diagnostics

You can diagnose RSC by using the Windows PowerShell cmdlets **Get-NetAdapterRsc** and **Get-NetAdapterStatistics**.

Following is example output when you run the Get-NetAdapterRsc cmdlet.

```
PS C:\Users\Administrator> Get-NetAdapterRsc

Name                      IPv4Enabled  IPv6Enabled  IPv4Operational IPv6Operational
IPv4FailureReason             IPv6Failure
                                       Reason
----                          -----------  -----------  --------------- --------------- ----------------- -
-----------
Ethernet                      True         False        True            False                 NoFailure
NicProperties
```

The **Get** cmdlet shows whether RSC is enabled in the interface and whether TCP enables RSC to be in an operational state. The failure reason provides details about the failure to enable RSC on that interface.

In the previous scenario, IPv4 RSC is supported and operational in the interface. To understand diagnostic failures, one can see the coalesced bytes or exceptions caused. This provides an indication of the coalescing issues.

Following is example output when you run the Get-NetAdapterStatistics cmdlet.

```
PS C:\Users\Administrator> $x = Get-NetAdapterStatistics "myAdapter"
PS C:\Users\Administrator> $x.rscstatistics

CoalescedBytes       : 0
CoalescedPackets     : 0
CoalescingEvents     : 0
CoalescingExceptions : 0
```

RSC and Virtualization

RSC is only supported in the physical host when the host network adapter is not bound to the Hyper-V Virtual Switch. RSC is disabled by the operating system when the host is bound to the Hyper-V Virtual Switch. In addition, virtual machines do not get the benefit of RSC because virtual network adapters do not support RSC.

RSC can be enabled for a virtual machine when Single Root Input/Output Virtualization (SR-IOV) is enabled. In this case, virtual functions support RSC capability; hence, virtual machines also receive the benefit of RSC.

Network Adapter Resources

A few network adapters actively manage their resources to achieve optimum performance. Several network adapters allow you to manually configure resources by using the **Advanced Networking** tab for the adapter. For such adapters, you can set the values of a number of parameters, including the number of receive buffers and send buffers.

Configuring network adapter resources is simplified by the use of the following Windows PowerShell cmdlets.

- Get-NetAdapterAdvancedProperty

- Set-NetAdapterAdvancedProperty
- Enable-NetAdapter
- Enable-NetAdapterBinding
- Enable-NetAdapterChecksumOffload
- Enable-NetAdapterIPSecOffload
- Enable-NetAdapterLso
- Enable-NetAdapterPowerManagement
- Enable-NetAdapterQos
- Enable-NetAdapterRDMA
- Enable-NetAdapterSriov

For more information, see Network Adapter Cmdlets in Windows PowerShell.

For links to all topics in this guide, see Network Subsystem Performance Tuning.

Configure the Order of Network Interfaces

12/16/2022 • 2 minutes to read • Edit Online

> Applies to: Windows Server 2022, Windows Server 2019, Windows Server 2016, Azure Stack HCI, versions 21H2 and 20H2

In Windows Server 2016 and Windows 10, you can use the interface metric to configure the order of network interfaces.

This is different than in previous versions of Windows and Windows Server, which allowed you to configure the binding order of network adapters by using either the user interface or the commands **INetCfgComponentBindings::MoveBefore** and **INetCfgComponentBindings::MoveAfter**. These two methods for ordering network interfaces are not available in Windows Server 2016 and Windows 10.

Instead, you can use the new method for setting the enumerated order of network adapters by configuring the interface metric of each adapter. You can configure the interface metric by using the Set-NetIPInterface Windows PowerShell command.

When network traffic routes are chosen and you have configured the **InterfaceMetric** parameter of the **Set-NetIPInterface** command, the overall metric that is used to determine the interface preference is the sum of the route metric and the interface metric. Typically, the interface metric gives preference to a particular interface, such as using wired if both wired and wireless are available.

The following Windows PowerShell command example shows use of this parameter.

```
Set-NetIPInterface -InterfaceIndex 12 -InterfaceMetric 15
```

The order in which adapters appear in a list is determined by the IPv4 or IPv6 interface metric. For more information, see GetAdaptersAddresses function.

For links to all topics in this guide, see Network Subsystem Performance Tuning.

Performance Tuning Network Adapters

12/16/2022 • 14 minutes to read • Edit Online

Applies to: Windows Server 2022, Windows Server 2019, Windows Server 2016, Azure Stack HCI, versions 21H2 and 20H2

Use the information in this topic to tune the performance network adapters for computers that are running Windows Server 2016 and later versions. If your network adapters provide tuning options, you can use these options to optimize network throughput and resource usage.

The correct tuning settings for your network adapters depend on the following variables:

- The network adapter and its feature set
- The type of workload that the server performs
- The server hardware and software resources
- Your performance goals for the server

The following sections describe some of your performance tuning options.

Enabling offload features

Turning on network adapter offload features is usually beneficial. However, the network adapter might not be powerful enough to handle the offload capabilities with high throughput.

> **IMPORTANT**
> Do not use the offload features **IPsec Task Offload** or **TCP Chimney Offload**. These technologies are deprecated in Windows Server 2016, and might adversely affect server and networking performance. In addition, these technologies might not be supported by Microsoft in the future.

For example, consider a network adapter that has limited hardware resources. In that case, enabling segmentation offload features might reduce the maximum sustainable throughput of the adapter. However, if the reduced throughput is acceptable, you should go ahead an enable the segmentation offload features.

> **NOTE**
> Some network adapters require you to enable offload features independently for the send and receive paths.

Enabling receive-side scaling (RSS) for web servers

RSS can improve web scalability and performance when there are fewer network adapters than logical processors on the server. When all the web traffic is going through the RSS-capable network adapters, the server can process incoming web requests from different connections simultaneously across different CPUs.

> **IMPORTANT**
>
> Avoid using both non-RSS network adapters and RSS-capable network adapters on the same server. Because of the load distribution logic in RSS and Hypertext Transfer Protocol (HTTP), performance might be severely degraded if a non-RSS-capable network adapter accepts web traffic on a server that has one or more RSS-capable network adapters. In this circumstance, you should use RSS-capable network adapters or disable RSS on the network adapter properties **Advanced Properties** tab.
>
> To determine whether a network adapter is RSS-capable, you can view the RSS information on the network adapter properties **Advanced Properties** tab.

RSS Profiles and RSS Queues

The default RSS predefined profile is **NUMAStatic**, which differs from the default that the previous versions of Windows used. Before you start using RSS profiles, review the available profiles to understand when they are beneficial and how they apply to your network environment and hardware.

For example, if you open Task Manager and review the logical processors on your server, and they seem to be underutilized for receive traffic, you can try increasing the number of RSS queues from the default of two to the maximum that your network adapter supports. Your network adapter might have options to change the number of RSS queues as part of the driver.

Increasing network adapter resources

For network adapters that allow you to manually configure 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 the receive buffer value to the maximum.

> **NOTE**
>
> If a network adapter does not expose manual resource configuration, either it dynamically configures the resources, or the resources are set to a fixed value that cannot be changed.

Enabling interrupt moderation

To control interrupt moderation, some network adapters expose different interrupt moderation levels, different buffer coalescing parameters (sometimes separately for send and receive buffers), or both.

You should consider interrupt moderation for CPU-bound workloads. When using interrupt moderation, 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 it does expose buffer coalescing, you can improve performance by increasing the number of coalesced buffers to allow more buffers per send or receive.

Performance tuning for low-latency packet processing

Many network adapters provide options to optimize operating system-induced latency. Latency is the elapsed time between the network driver processing an incoming packet and the network driver sending the packet back. This time is usually measured in microseconds. For comparison, the transmission time for packet transmissions over long distances is usually measured in milliseconds (an order of magnitude larger). This tuning will not reduce the time a packet spends in transit.

Following are some performance tuning suggestions for microsecond-sensitive networks.

- Set the computer BIOS to **High Performance**, with C-states disabled. However, note that this is system and BIOS dependent, and some systems will provide higher performance if the operating system controls power management. You can check and adjust your power management settings from **Settings** or by using the **powercfg** command. For more information, see Powercfg Command-Line Options.
- Set the operating system power management profile to **High Performance System**.

  > **NOTE**
  > This setting does not work properly if the system BIOS has been set to disable operating system control of power management.

- Enable static offloads. For example, enable the UDP Checksums, TCP Checksums, and Send Large Offload (LSO) settings.
- If the traffic is multi-streamed, such as when receiving high-volume multicast traffic, enable RSS.
- Disable the **Interrupt Moderation** setting for network card drivers that require the lowest possible latency. Remember, this configuration can use more CPU time and it represents a tradeoff.
- Handle network adapter interrupts and DPCs on a core processor that shares CPU cache with the core that is being used by the program (user thread) that is handling the packet. CPU affinity tuning can be used to direct a process to certain logical processors in conjunction with RSS configuration to accomplish this. Using the same core for the interrupt, DPC, and user mode thread exhibits worse performance as load increases because the ISR, DPC, and thread contend for the use of the core.

System management interrupts

Many hardware systems use System Management Interrupts (SMI) for a variety of maintenance functions, such as reporting error correction code (ECC) memory errors, maintaining legacy USB compatibility, controlling the fan, and managing BIOS-controlled power settings.

The SMI is the highest-priority interrupt on the system, and places the CPU in a management mode. This mode preempts all other activity while SMI runs an interrupt service routine, typically contained in BIOS.

Unfortunately, this behavior can result in latency spikes of 100 microseconds or more.

If you need to achieve the lowest latency, you should request a BIOS version from your hardware provider that reduces SMIs to the lowest degree possible. These BIOS versions are frequently referred to as "low latency BIOS" or "SMI free BIOS." In some cases, it is not possible for a hardware platform to eliminate SMI activity altogether because it is used to control essential functions (for example, cooling fans).

> **NOTE**
> The operating system cannot control SMIs because the logical processor is running in a special maintenance mode, which prevents operating system intervention.

Performance tuning TCP

You can use the following items to tune TCP performance.

TCP receive window autotuning

In Windows Vista, Windows Server 2008, and later versions of Windows, the Windows network stack uses a feature that is named *TCP receive window autotuning level* to negotiate the TCP receive window size. This feature can negotiate a defined receive window size for every TCP communication during the TCP Handshake.

In earlier versions of Windows, the Windows network stack used a fixed-size receive window (65,535 bytes) that limited the overall potential throughput for connections. The total achievable throughput of TCP connections could limit network usage scenarios. TCP receive window autotuning enables these scenarios to fully use the network.

For a TCP receive window that has a particular size, you can use the following equation to calculate the total throughput of a single connection.

> *Total achievable throughput in bytes = TCP receive window size in bytes* * (1 / *connection latency in seconds*)

For example, for a connection that has a latency of 10 ms, the total achievable throughput is only 51 Mbps. This value is reasonable for a large corporate network infrastructure. However, by using autotuning to adjust the receive window, the connection can achieve the full line rate of a 1-Gbps connection.

Some applications define the size of the TCP receive window. If the application does not define the receive window size, the link speed determines the size as follows:

- Less than 1 megabit per second (Mbps): 8 kilobytes (KB)
- 1 Mbps to 100 Mbps: 17 KB
- 100 Mbps to 10 gigabits per second (Gbps): 64 KB
- 10 Gbps or faster: 128 KB

For example, on a computer that has a 1-Gbps network adapter installed, the window size should be 64 KB.

This feature also makes full use of other features to improve network performance. These features include the rest of the TCP options that are defined in RFC 1323. By using these features, Windows-based computers can negotiate TCP receive window sizes that are smaller but are scaled at a defined value, depending on the configuration. This behavior the sizes easier to handle for networking devices.

> **NOTE**
>
> You may experience an issue in which the network device is not compliant with the **TCP window scale option**, as defined in RFC 1323 and, therefore, doesn't support the scale factor. In such cases, refer to this KB 934430, Network connectivity fails when you try to use Windows Vista behind a firewall device or contact the Support team for your network device vendor.

Review and configure TCP receive window autotuning level

You can use either netsh commands or Windows PowerShell cmdlets to review or modify the TCP receive window autotuning level.

> **NOTE**
>
> Unlike in versions of Windows that pre-date Windows 10 or Windows Server 2019, you can no longer use the registry to configure the TCP receive window size. For more information about the deprecated settings, see Deprecated TCP parameters.

> **NOTE**
>
> For detailed information about the available autotuning levels, see Autotuning levels.

To use netsh to review or modify the autotuning level

To review the current settings, open a Command Prompt window and run the following command:

```
netsh interface tcp show global
```

The output of this command should resemble the following:

```
Querying active state...

TCP Global Parameters
-----
Receive-Side Scaling State : enabled
Chimney Offload State : disabled
Receive Window Auto-Tuning Level : normal
Add-On Congestion Control Provider : default
ECN Capability : disabled
RFC 1323 Timestamps : disabled
Initial RTO : 3000
Receive Segment Coalescing State : enabled
Non Sack Rtt Resiliency : disabled
Max SYN Retransmissions : 2
Fast Open : enabled
Fast Open Fallback : enabled
Pacing Profile : off
```

To modify the setting, run the following command at the command prompt:

```
netsh interface tcp set global autotuninglevel=<Value>
```

> **NOTE**
>
> In the preceding command, <*Value*> represents the new value for the auto tuning level.

For more information about this command, see Netsh commands for Interface Transmission Control Protocol.

To use Powershell to review or modify the autotuning level

To review the current settings, open a PowerShell window and run the following cmdlet.

```
Get-NetTCPSetting | Select SettingName,AutoTuningLevelLocal
```

The output of this cmdlet should resemble the following.

```
SettingName           AutoTuningLevelLocal
-----------          --------------------
Automatic
InternetCustom       Normal
DatacenterCustom     Normal
Compat               Normal
Datacenter           Normal
Internet             Normal
```

To modify the setting, run the following cmdlet at the PowerShell command prompt.

```
Set-NetTCPSetting -AutoTuningLevelLocal <Value>
```

> **NOTE**
> In the preceding command, < *Value*> represents the new value for the auto tuning level.

For more information about these cmdlets, see the following articles:

- Get-NetTCPSetting
- Set-NetTCPSetting

Autotuning levels

You can set receive window autotuning to any of five levels. The default level is **Normal**. The following table describes the levels.

LEVEL	HEXADECIMAL VALUE	COMMENTS
Normal (default)	0x8 (scale factor of 8)	Set the TCP receive window to grow to accommodate almost all scenarios.
Disabled	No scale factor available	Set the TCP receive window at its default value.
Restricted	0x4 (scale factor of 4)	Set the TCP receive window to grow beyond its default value, but limit such growth in some scenarios.
Highly Restricted	0x2 (scale factor of 2)	Set the TCP receive window to grow beyond its default value, but do so very conservatively.
Experimental	0xE (scale factor of 14)	Set the TCP receive window to grow to accommodate extreme scenarios.

If you use an application to capture network packets, the application should report data that resembles the following for different window autotuning level settings.

- Autotuning level: **Normal (default state)**

```
Frame: Number = 492, Captured Frame Length = 66, MediaType = ETHERNET
+ Ethernet: Etype = Internet IP (IPv4),DestinationAddress:[D8-FE-E3-65-F3-FD],SourceAddress:[C8-5B-
76-7D-FA-7F]
+ Ipv4: Src = 192.169.0.5, Dest = 192.169.0.4, Next Protocol = TCP, Packet ID = 2667, Total IP Length
= 52
- Tcp: [Bad CheckSum]Flags=......S., SrcPort=60975, DstPort=Microsoft-DS(445), PayloadLen=0,
Seq=4075590425, Ack=0, Win=64240 ( Negotiating scale factor 0x8 ) = 64240
SrcPort: 60975
DstPort: Microsoft-DS(445)
SequenceNumber: 4075590425 (0xF2EC9319)
AcknowledgementNumber: 0 (0x0)
+ DataOffset: 128 (0x80)
+ Flags: ......S. ----------------------------------------------------------> SYN Flag set
Window: 64240 ( Negotiating scale factor 0x8 ) = 64240 ---------> TCP Receive Window set as 64K as
per NIC Link bitrate. Note it shows the 0x8 Scale Factor.
Checksum: 0x8182, Bad
UrgentPointer: 0 (0x0)
- TCPOptions:
+ MaxSegmentSize: 1
+ NoOption:
+ WindowsScaleFactor: ShiftCount: 8 ----------------------------> Scale factor, defined by
AutoTuningLevel
+ NoOption:
+ NoOption:
+ SACKPermitted:
```

- Autotuning level: **Disabled**

```
Frame: Number = 353, Captured Frame Length = 62, MediaType = ETHERNET
+ Ethernet: Etype = Internet IP (IPv4),DestinationAddress:[D8-FE-E3-65-F3-FD],SourceAddress:[C8-5B-
76-7D-FA-7F]
+ Ipv4: Src = 192.169.0.5, Dest = 192.169.0.4, Next Protocol = TCP, Packet ID = 2576, Total IP Length
= 48
- Tcp: [Bad CheckSum]Flags=......S., SrcPort=60956, DstPort=Microsoft-DS(445), PayloadLen=0,
Seq=2315885330, Ack=0, Win=64240 ( ) = 64240
SrcPort: 60956
DstPort: Microsoft-DS(445)
SequenceNumber: 2315885330 (0x8A099B12)
AcknowledgementNumber: 0 (0x0)
+ DataOffset: 112 (0x70)
+ Flags: ......S. ----------------------------------------------------------> SYN Flag set
Window: 64240 ( ) = 64240 ---------------------------------------> TCP Receive Window set as 64K as
per NIC Link bitrate. Note there is no Scale Factor defined. In this case, Scale factor is not being
sent as a TCP Option, so it will not be used by Windows.
Checksum: 0x817E, Bad
UrgentPointer: 0 (0x0)
- TCPOptions:
+ MaxSegmentSize: 1
+ NoOption:
+ NoOption:
+ SACKPermitted:
```

- Autotuning level: **Restricted**

```
Frame: Number = 3, Captured Frame Length = 66, MediaType = ETHERNET
+ Ethernet: Etype = Internet IP (IPv4),DestinationAddress:[D8-FE-E3-65-F3-FD],SourceAddress:[C8-5B-
76-7D-FA-7F]
+ Ipv4: Src = 192.169.0.5, Dest = 192.169.0.4, Next Protocol = TCP, Packet ID = 2319, Total IP Length
= 52
- Tcp: [Bad CheckSum]Flags=......S., SrcPort=60890, DstPort=Microsoft-DS(445), PayloadLen=0,
Seq=1966088568, Ack=0, Win=64240 ( Negotiating scale factor 0x4 ) = 64240
SrcPort: 60890
DstPort: Microsoft-DS(445)
SequenceNumber: 1966088568 (0x75302178)
AcknowledgementNumber: 0 (0x0)
+ DataOffset: 128 (0x80)
+ Flags: ......S. ----------------------------------------------------------> SYN Flag set
Window: 64240 ( Negotiating scale factor 0x4 ) = 64240 ---------> TCP Receive Window set as 64K as
per NIC Link bitrate. Note it shows the 0x4 Scale Factor.
Checksum: 0x8182, Bad
UrgentPointer: 0 (0x0)
- TCPOptions:
+ MaxSegmentSize: 1
+ NoOption:
+ WindowsScaleFactor: ShiftCount: 4 ------------------------------> Scale factor, defined by
AutoTuningLevel.
+ NoOption:
+ NoOption:
+ SACKPermitted:
```

- Autotuning level: **Highly restricted**

```
Frame: Number = 115, Captured Frame Length = 66, MediaType = ETHERNET
+ Ethernet: Etype = Internet IP (IPv4),DestinationAddress:[D8-FE-E3-65-F3-FD],SourceAddress:[C8-5B-
76-7D-FA-7F]
+ Ipv4: Src = 192.169.0.5, Dest = 192.169.0.4, Next Protocol = TCP, Packet ID = 2388, Total IP Length
= 52
- Tcp: [Bad CheckSum]Flags=......S., SrcPort=60903, DstPort=Microsoft-DS(445), PayloadLen=0,
Seq=1463725706, Ack=0, Win=64240 ( Negotiating scale factor 0x2 ) = 64240
SrcPort: 60903
DstPort: Microsoft-DS(445)
SequenceNumber: 1463725706 (0x573EAE8A)
AcknowledgementNumber: 0 (0x0)
+ DataOffset: 128 (0x80)
+ Flags: ......S. ----------------------------------------------------------> SYN Flag set
Window: 64240 ( Negotiating scale factor 0x2 ) = 64240 ---------> TCP Receive Window set as 64K as
per NIC Link bitrate. Note it shows the 0x2 Scale Factor.
Checksum: 0x8182, Bad
UrgentPointer: 0 (0x0)
- TCPOptions:
+ MaxSegmentSize: 1
+ NoOption:
+ WindowsScaleFactor: ShiftCount: 2 ------------------------------> Scale factor, defined by
AutoTuningLevel
+ NoOption:
+ NoOption:
+ SACKPermitted:
```

- Autotuning level: **Experimental**

```
Frame: Number = 238, Captured Frame Length = 66, MediaType = ETHERNET
+ Ethernet: Etype = Internet IP (IPv4),DestinationAddress:[D8-FE-E3-65-F3-FD],SourceAddress:[C8-5B-
76-7D-FA-7F]
+ Ipv4: Src = 192.169.0.5, Dest = 192.169.0.4, Next Protocol = TCP, Packet ID = 2490, Total IP Length
= 52
- Tcp: [Bad CheckSum]Flags=......S., SrcPort=60933, DstPort=Microsoft-DS(445), PayloadLen=0,
Seq=2095111365, Ack=0, Win=64240 ( Negotiating scale factor 0xe ) = 64240
SrcPort: 60933
DstPort: Microsoft-DS(445)
SequenceNumber: 2095111365 (0x7CE0DCC5)
AcknowledgementNumber: 0 (0x0)
+ DataOffset: 128 (0x80)
+ Flags: ......S. ----------------------------------------------------------> SYN Flag set
Window: 64240 ( Negotiating scale factor 0xe ) = 64240 ---------> TCP Receive Window set as 64K as
per NIC Link bitrate. Note it shows the 0xe Scale Factor.
Checksum: 0x8182, Bad
UrgentPointer: 0 (0x0)
- TCPOptions:
+ MaxSegmentSize: 1
+ NoOption:
+ WindowsScaleFactor: ShiftCount: 14 ----------------------------> Scale factor, defined by
AutoTuningLevel
+ NoOption:
+ NoOption:
+ SACKPermitted:
```

Deprecated TCP parameters

The following registry settings from Windows Server 2003 are no longer supported, and are ignored in later versions.

- TcpWindowSize
- NumTcbTablePartitions
- MaxHashTableSize

All of these settings were located in the following registry subkey:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters
```

Windows Filtering Platform

Windows Vista and Windows Server 2008 introduced the Windows Filtering Platform (WFP). WFP provides APIs to non-Microsoft independent software vendors (ISVs) to create packet processing filters. Examples include firewall and antivirus software.

> **NOTE**
>
> A poorly-written WFP filter can significantly decrease a server's networking performance. For more information, see Porting Packet-Processing Drivers and Apps to WFP in the Windows Dev Center.

For links to all topics in this guide, see Network Subsystem Performance Tuning.

Network-Related Performance Counters

12/16/2022 • 2 minutes to read • Edit Online

> Applies to: Windows Server 2022, Windows Server 2019, Windows Server 2016, Azure Stack HCI, versions 21H2 and 20H2

This topic lists the counters that are relevant to managing network performance, and contains the following sections.

Resource Utilization

The following performance counters are relevant to network resource utilization.

- IPv4, IPv6
  - Datagrams Received/sec
  - Datagrams Sent/sec
- TCPv4, TCPv6
  - Segments Received/sec
  - Segments Sent/sec
  - Segments Retransmitted/sec
- Network Interface(*), Network Adapter(*)
  - Bytes Received/sec
  - Bytes Sent/sec
  - Packets Received/sec
  - Packets Sent/sec
  - 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.
- Processor Information
  - % Processor Time
  - Interrupts/sec
  - DPCs Queued/sec

    This counter is an average rate at which DPCs were added to the logical processor's DPC queue.

Each logical processor has its own DPC queue. This counter measures the rate at which 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.

Potential Network Problems

The following performance counters are relevant to potential network problems.

- Network Interface(*), Network Adapter(*)
  - Packets Received Discarded
  - Packets Received Errors
  - Packets Outbound Discarded
  - Packets Outbound Errors
- WFPv4, WFPv6
  - Packets Discarded/sec
- UDPv4, UDPv6
  - Datagrams Received Errors
- TCPv4, TCPv6
  - Connection Failures
  - Connections Reset
- Network QoS Policy
  - Packets dropped
  - Packets dropped/sec
- Per Processor Network Interface Card Activity
  - Low Resource Receive Indications/sec
  - Low Resource Received Packets/sec
- Microsoft Winsock BSP
  - Dropped Datagrams
  - Dropped Datagrams/sec
  - Rejected Connections
  - Rejected Connections/sec

Receive Side Coalescing (RSC) performance

The following performance counters are relevant to RSC performance.

- Network Adapter(*)
  - TCP Active RSC Connections
  - TCP RSC Average Packet Size
  - TCP RSC Coalesced Packets/sec

- TCP RSC Exceptions/sec

For links to all topics in this guide, see Network Subsystem Performance Tuning.

Performance Tools for Network Workloads

12/16/2022 • 2 minutes to read • Edit Online

> Applies to: Windows Server 2022, Windows Server 2019, Windows Server 2016, Azure Stack HCI, versions 21H2 and 20H2

You can use this topic to learn about performance tools.

This topic contains sections about the Client to Server Traffic tool and TCP/IP Window Size.

Client to Server Traffic tool

The Client to Server Traffic (ctsTraffic) tool provides you with the ability to create and verify network traffic.

For more information, and to download the tool, see ctsTraffic (Client-To-Server Traffic).

TCP/IP Window Size

For 1 GB adapters, the settings shown in the previous table should provide good throughput because NTttcp sets the default TCP window size to 64 K through a specific logical processor 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, the default TCP window size value for NTttcp 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 using it 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.

For more information, see Network Subsystem Performance Tuning.

Performance Tuning Software Defined Networks

12/16/2022 • 3 minutes to read • Edit Online

Software Defined Networking (SDN) in Windows Server 2016 is made up of a combination of a Network Controller, Hyper-V Hosts, Software Load Balancer Gateways and HNV Gateways. For tuning of each of these components refer to the following sections:

Network Controller

The network controller is a Windows Server role which must be enabled on Virtual Machines running on hosts that are configured to use SDN and are controlled by the network controller.

Three Network Controller enabled VMs are sufficient for high availability and maximum performance. Each VM must be sized according to the guidelines provided in the SDN infrastructure virtual machine role requirements section of the Plan a Software Defined Network Infrastructure topic.

SDN Quality of Service (QoS)

To ensure virtual machine traffic is prioritized effectively and fairly, it is recommended that you configure SDN QoS on the workload virtual machines. For more information on configuring SDN QoS, refer to the Configure QoS for a Tenant VM Network Adapter topic.

Hyper-V Host Networking

The guidance provided in the Hyper-V network I/O performance section of the Performance Tuning for Hyper-V Servers guide is applicable when SDN is used, however this section covers additional guidelines that must be followed to ensure the best performance when using SDN.

Physical Network Adapter (NIC) Teaming

For best performance and fail-over capabilities, it is recommended that you configure the physical network adapters to be teamed. When using SDN you must create the team with Switch Embedded Teaming (SET).

The optimal number of team members is two as virtualized traffic will be spread across both of the team members for both inbound and outbound directions. You can have more than two team members; however inbound traffic will be spread over at most two of the adapters. Outbound traffic will always be spread across all adapters if the default of dynamic load balancing remains configured on the virtual switch.

Encapsulation Offloads

SDN relies on encapsulation of packets to virtualize the network. For optimal performance, it is important that the network adapter supports hardware offload for the encapsulation format that is used. There is no significant performance benefit of one encapsulation format over another. The default encapsulation format when the network controller is used is VXLAN.

You can determine which encapsulation format is being used through the network controller with the following PowerShell cmdlet:

```
    (Get-NetworkControllerVirtualNetworkConfiguration -connectionuri
$uri).properties.networkvirtualizationprotocol
```

For best performance, if VXLAN is returned then you must make sure your physical network adapters support VXLAN task offload. If NVGRE is returned, then your physical network adapters must support NVGRE task offload.

MTU

Encapsulation results in extra bytes being added to each packet. In order to avoid fragmentation of these packets, the physical network must be configured to use jumbo frames. An MTU value of 9234 is the recommended size for either VXLAN or NVGRE and must be configured on the physical switch for the physical interfaces of the host ports (L2) and the router interfaces (L3) of the VLANs over which encapsulated packets will be sent. This includes the Transit, HNV Provider and Management networks.

MTU on the Hyper-V host is configured through the network adapter, and the Network Controller Host Agent running on the Hyper-V host will adjust for the encapsulation overhead automatically if supported by the network adapter driver.

Once traffic egresses from the virtual network via a Gateway, the encapsulation is removed and the original MTU as sent from the VM is used.

Single Root IO Virtualization (SR-IOV)

SDN is implemented on the Hyper-V host using a forwarding switch extension in the virtual switch. For this switch extension to process packets, SR-IOV must not be used on virtual network interfaces that are configured for use with the network controller as it causes VM traffic to bypass the virtual switch.

SR-IOV can still be enabled on the virtual switch if desired and can be used by VM network adapters that are not controlled by the network controller. These SR-IOV VMs can coexist on the same virtual switch as network controller controlled VMs which do not use SR-IOV.

If you are using 40Gbit network adapters it is recommended that you enable SR-IOV on the virtual switch for the Software Load Balancing (SLB) Gateways to achieve maximum throughput. This is covered in more detail in the Software Load Balancer Gateways section.

HNV Gateways

You can find information on tuning HNV Gateways for use with SDN in the HVN Gateways section.

Software Load Balancer (SLB)

SLB Gateways can only be used with the Network Controller and SDN. You can find more information on tuning SDN for use with SLB Gateways in the Software Load Balancer Gateways section.

HNV Gateway Performance Tuning in Software Defined Networks

12/16/2022 • 6 minutes to read • Edit Online

This topic provides hardware specifications and configuration recommendations for servers that are running Hyper-V and hosting Windows Server Gateway virtual machines, in addition to configuration parameters for Windows Server Gateway virtual machines (VMs). To extract best performance from Windows Server gateway VMs, it is expected that these guidelines will be followed. The following sections contain hardware and configuration requirements when you deploy Windows Server Gateway.

1. Hyper-V hardware recommendations
2. Hyper-V host configuration
3. Windows Server gateway VM configuration

Hyper-V hardware recommendations

Following is the recommended minimum hardware configuration for each server that is running Windows Server 2016 and Hyper-V.

SERVER COMPONENT	SPECIFICATION
Central Processing Unit (CPU)	Non-Uniform Memory Architecture (NUMA) nodes: 2 If there are multiple Windows Server gateway VMs on the host, for best performance, each gateway VM should have full access to one NUMA node. And it should be different from the NUMA node used by the host physical adapter.
Cores per NUMA node	2
Hyper-Threading	Disabled. Hyper-Threading does not improve the performance of Windows Server Gateway.
Random Access Memory (RAM)	48 GB
Network Interface Cards (NICs)	Two 10 GB NICs,The gateway performance will depend on the line rate. If the line rate is less than 10Gbps, the gateway tunnel throughput numbers will also go down by the same factor.

Ensure that the number of virtual processors that are assigned to a Windows Server Gateway VM does not exceed the number of processors on the NUMA node. For example, if a NUMA node has 8 cores, the number of virtual processors should be less than or equal to 8. For best performance, it should be 8. To find out the number of NUMA nodes and the number of cores per NUMA node, run the following Windows PowerShell script on each Hyper-V host:

```
$nodes = [object[]] $(gwmi –Namespace root\virtualization\v2 -Class MSVM_NumaNode)
$cores = ($nodes | Measure-Object NumberOfProcessorCores -sum).Sum
$lps = ($nodes | Measure-Object NumberOfLogicalProcessors -sum).Sum


Write-Host "Number of NUMA Nodes: ", $nodes.count
Write-Host ("Total Number of Cores: ", $cores)
Write-Host ("Total Number of Logical Processors: ", $lps)
```

IMPORTANT

Allocating virtual processors across NUMA nodes might have a negative performance impact on Windows Server Gateway. Running multiple VMs, each of which has virtual processors from one NUMA node, likely provides better aggregate performance than a single VM to which all virtual processors are assigned.

One gateway VM with eight virtual processors and at least 8GB RAM is recommended when selecting the number of gateway VMs to install on each Hyper-V host when each NUMA node has eight cores. In this case, one NUMA node is dedicated to the host machine.

Hyper-V Host configuration

Following is the recommended configuration for each server that is running Windows Server 2016 and Hyper-V and whose workload is to run Windows Server Gateway VMs. These configuration instructions include the use of Windows PowerShell command examples. These examples contain placeholders for actual values that you need to provide when you run the commands in your environment. For example, network adapter name placeholders are "NIC1" and "NIC2." When you run commands that use these placeholders, utilize the actual names of the network adapters on your servers rather than using the placeholders, or the commands will fail.

NOTE

To run the following Windows PowerShell commands, you must be a member of the Administrators group.

CONFIGURATION ITEM	WINDOWS POWERSHELL CONFIGURATION
Switch Embedded Teaming	When you create a vswitch with multiple network adapters, it automatically enabled switch embedded teaming for those adapters. `New-VMSwitch -Name TeamedvSwitch -NetAdapterName "NIC 1","NIC 2"` Traditional teaming through LBFO is not supported with SDN in Windows Server 2016. Switch Embedded Teaming allows you to use the same set of NICs for your virtual traffic and RDMA traffic. This was not supported with NIC teaming based on LBFO.
Interrupt Moderation on physical NICs	Use default settings. To check the configuration, you can use the following Windows PowerShell command: `Get-NetAdapterAdvancedProperty`

CONFIGURATION ITEM	WINDOWS POWERSHELL CONFIGURATION
Receive Buffers size on physical NICs	You can verify whether the physical NICs support the configuration of this parameter by running the command `Get-NetAdapterAdvancedProperty`. If they do not support this parameter, the output from the command does not include the property "Receive Buffers." If NICs do support this parameter, you can use the following Windows PowerShell command to set the Receive Buffers size: `Set-NetAdapterAdvancedProperty "NIC1" –DisplayName "Receive Buffers" –DisplayValue 3000`
Send Buffers size on physical NICs	You can verify whether the physical NICs support the configuration of this parameter by running the command `Get-NetAdapterAdvancedProperty`. If the NICs do not support this parameter, the output from the command does not include the property "Send Buffers." If NICs do support this parameter, you can use the following Windows PowerShell command to set the Send Buffers size: `Set-NetAdapterAdvancedProperty "NIC1" –DisplayName "Transmit Buffers" –DisplayValue 3000`

Receive Side Scaling (RSS) on physical NICs	You can verify whether your physical NICs have RSS enabled by running the Windows PowerShell command Get-NetAdapterRss. You can use the following Windows PowerShell commands to enable and configure RSS on your network adapters:
	`Enable-NetAdapterRss "NIC1","NIC2"`
	`Set-NetAdapterRss "NIC1","NIC2" –NumberOfReceiveQueues 16 -MaxProcessors`
	NOTE: If VMMQ or VMQ is enabled, RSS does not have to be enabled on the physical network adapters. You can enable it on the host virtual network adapters

VMMQ	To enable VMMQ for a VM, run the following command: `Set-VmNetworkAdapter -VMName <gateway vm name>,-VrssEnabled $true -VmmqEnabled $true` NOTE: Not all network adapters support VMMQ. Currently, it is supported on Chelsio T5 and T6, Mellanox CX-3 and CX-4, and QLogic 45xxx series
Virtual Machine Queue (VMQ) on the NIC Team	You can enable VMQ on your SET team by using the following Windows PowerShell command: `Enable-NetAdapterVmq` NOTE: This should be enabled only if the HW does not support VMMQ. If supported, VMMQ should be enabled for better performance.

NOTE

VMQ and vRSS come into picture only when the load on the VM is high and the CPU is being utilized to the maximum. Only then will at least one processor core max out. VMQ and vRSS will then be beneficial to help spread the processing load across multiple cores. This is not applicable for IPsec traffic as IPsec traffic is confined to a single core.

Windows Server Gateway VM configuration

On both Hyper-V hosts, you can configure multiple VMs that are configured as gateways with Windows Server Gateway. You can use Virtual Switch Manager to create a Hyper-V Virtual Switch that is bound to the NIC team on the Hyper-V host. Note that for best performance, you should deploy a single gateway VM on a Hyper-V host.

Following is the recommended configuration for each Windows Server Gateway VM.

CONFIGURATION ITEM	WINDOWS POWERSHELL CONFIGURATION
Memory	8 GB
Number of virtual network adapters	3 NICs with the following specific uses: 1 for Management that is used by the management operating system, 1 External that provides access to external networks, 1 that is Internal that provides access to internal networks only.
Receive Side Scaling (RSS)	You can keep the default RSS settings for the Management NIC. The following example configuration is for a VM that has 8 virtual processors. For the External and Internal NICs, you can enable RSS with BaseProcNumber set to 0 and MaxRssProcessors set to 8 using the following Windows PowerShell command: `Set-NetAdapterRss "Internal","External" –BaseProcNumber 0 –MaxProcessorNumber 8`
Send side buffer	You can keep the default Send Side Buffer settings for the Management NIC. For both the Internal and External NICs you can configure the Send Side Buffer with 32 MB of RAM by using the following Windows PowerShell command: `Set-NetAdapterAdvancedProperty "Internal","External" –DisplayName "Send Buffer Size" –DisplayValue "32MB"`
Receive Side buffer	You can keep the default Receive Side Buffer settings for the Management NIC. For both the Internal and External NICs, you can configure the Receive Side Buffer with 16 MB of RAM by using the following Windows PowerShell command: `Set-NetAdapterAdvancedProperty "Internal","External" –DisplayName "Receive Buffer Size" –DisplayValue "16MB"`
Forward Optimization	You can keep the default Forward Optimization settings for the Management NIC. For both the Internal and External NICs, you can enable Forward Optimization by using the following Windows PowerShell command: `Set-NetAdapterAdvancedProperty "Internal","External" –DisplayName "Forward Optimization" –DisplayValue "1"`

SLB Gateway Performance Tuning in Software Defined Networks

12/16/2022 • 2 minutes to read • Edit Online

Software load balancing is provided by a combination of a load balancer manager in the Network Controller VMs, the Hyper-V Virtual Switch and a set of Load Balancer Multixplexor (Mux) VMs.

No additional performance tuning is required to configure the Network Controller or the Hyper-V host for load balancing beyond what is described in the Software Defined Networking section, unless you will be using SR-IOV for the Muxes as described below.

SLB Mux VM Configuration

SLB Mux virtual machines are deployed in an Active-Active configuration. This means that every Mux VM that is deployed and added to the Network Controller can process incoming requests. Thus, the total aggregate throughput of all of the connections is only limited by the number of Mux VMs that you have deployed.

An individual connection to a Virtual IP (VIP) will always be sent to the same Mux, assuming the number of muxes remains constant, and as a result its throughput will be limited to the throughput of a single Mux VM. Muxes only process the inbound traffic that is destined to a VIP. Response packets go directly from the VM that is sending the response to the physical switch which forwards it on to the client.

In some cases when the source of the request originates from an SDN host that is added to the same Network Controller that manages the VIP, further optimization of the inbound path for the request is also performed which enables most packets to travel directly from the client to the server, bypassing the Mux VM entirely. No additional configuration is required for this optimization to take place.

Each SLB Mux VM must be sized according to the guidelines provided in the SDN infrastructure virtual machine role requirements section of the Plan a Software Defined Network Infrastructure topic.

Single Root IO virtualization (SR-IOV)

When using 40Gbit Ethernet, the ability for the virtual switch to process packets for the Mux VM becomes the limiting factor for Mux VM throughput. Because of this it is recommended that SR-IOV be enabled on the SLB VM's VM Network Adapter to ensure that the virtual switch is not the bottleneck.

To enable SR-IOV, you must enable it on the virtual switch when the virtual switch is created. In this example, we are creating a virtual switch with switch embedded teaming (SET) and SR-IOV:

```
    new-vmswitch -Name SDNSwitch -EnableEmbeddedTeaming $true -NetAdapterName @("NIC1", "NIC2") -EnableIOV
$true
```

Then, it must be enabled on the virtual network adapter(s) of the SLB Mux VM which process the data traffic. In this example, SR-IOV is being enabled on all adapters:

```
    get-vmnetworkadapter -VMName SLBMUX1 | set-vmnetworkadapter -IovWeight 50
```

Advanced Data Deduplication settings

12/16/2022 • 12 minutes to read • Edit Online

> Applies to: Windows Server 2022, Windows Server 2019, Windows Server 2016, Azure Stack HCI, versions 21H2 and 20H2

This document describes how to modify advanced Data Deduplication settings. For recommended workloads, the default settings should be sufficient. The main reason to modify these settings is to improve Data Deduplication's performance with other kinds of workloads.

Modifying Data Deduplication job schedules

The default Data Deduplication job schedules are designed to work well for recommended workloads and be as non-intrusive as possible (excluding the *Priority Optimization* job that is enabled for the **Backup** usage type). When workloads have large resource requirements, it is possible to ensure that jobs run only during idle hours, or to reduce or increase the amount of system resources that a Data Deduplication job is allowed to consume.

Changing a Data Deduplication schedule

Data Deduplication jobs are scheduled via Windows Task Scheduler and can be viewed and edited there under the path Microsoft\Windows\Deduplication. Data Deduplication includes several cmdlets that make scheduling easy.

- `Get-DedupSchedule` shows the current scheduled jobs.
- `New-DedupSchedule` creates a new scheduled job.
- `Set-DedupSchedule` modifies an existing scheduled job.
- `Remove-DedupSchedule` removes a scheduled job.

The most common reason for changing when Data Deduplication jobs run is to ensure that jobs run during off hours. The following step-by-step example shows how to modify the Data Deduplication schedule for a *sunny day* scenario: a hyper-converged Hyper-V host that is idle on weekends and after 7:00 PM on week nights. To change the schedule, run the following PowerShell cmdlets in an Administrator context.

1. Disable the scheduled hourly Optimization jobs.

```
Set-DedupSchedule -Name BackgroundOptimization -Enabled $false
Set-DedupSchedule -Name PriorityOptimization -Enabled $false
```

2. Remove the currently scheduled Garbage Collection and Integrity Scrubbing jobs.

```
Get-DedupSchedule -Type GarbageCollection | ForEach-Object { Remove-DedupSchedule -InputObject $_ }
Get-DedupSchedule -Type Scrubbing | ForEach-Object { Remove-DedupSchedule -InputObject $_ }
```

3. Create a nightly Optimization job that runs at 7:00 PM with high priority and all the CPUs and memory available on the system.

```
New-DedupSchedule -Name "NightlyOptimization" -Type Optimization -DurationHours 11 -Memory 100 -
Cores 100 -Priority High -Days @(1,2,3,4,5) -Start (Get-Date "2016-08-08 19:00:00")
```

> **NOTE**
> The *date* part of the `System.Datetime` provided to `-Start` is irrelevant (as long as it's in the past), but the *time* part specifies when the job should start.

4. Create a weekly Garbage Collection job that runs on Saturday starting at 7:00 AM with high priority and all the CPUs and memory available on the system.

```
New-DedupSchedule -Name "WeeklyGarbageCollection" -Type GarbageCollection -DurationHours 23 -Memory
100 -Cores 100 -Priority High -Days @(6) -Start (Get-Date "2016-08-13 07:00:00")
```

5. Create a weekly Integrity Scrubbing job that runs on Sunday starting at 7 AM with high priority and all the CPUs and memory available on the system.

```
New-DedupSchedule -Name "WeeklyIntegrityScrubbing" -Type Scrubbing -DurationHours 23 -Memory 100 -
Cores 100 -Priority High -Days @(0) -Start (Get-Date "2016-08-14 07:00:00")
```

Available job-wide settings

You can toggle the following settings for new or scheduled Data Deduplication jobs:

PARAMETER NAME	DEFINITION	ACCEPTED VALUES	WHY WOULD YOU WANT TO SET THIS VALUE?
Type	The type of the job that should be scheduled	• Optimization • GarbageCollection • Scrubbing	This value is required because it is the type of job that you want to schedule. This value cannot be changed after the task has been scheduled.
Priority	The system priority of the scheduled job	• High • Medium • Low	This value helps the system determine how to allocate CPU time. *High* will use more CPU time, *low* will use less.
Days	The days that the job is scheduled	An array of integers 0-6 representing the days of the week: • 0 = Sunday • 1 = Monday • 2 = Tuesday • 3 = Wednesday • 4 = Thursday • 5 = Friday • 6 = Saturday	Scheduled tasks have to run on at least one day.
Cores	The percentage of cores on the system that a job should use	Integers 0-100 (indicates a percentage)	To control what level of impact a job will have on the compute resources on the system

PARAMETER NAME	DEFINITION	ACCEPTED VALUES	WHY WOULD YOU WANT TO SET THIS VALUE?
DurationHours	The maximum number of hours a job should be allowed to run	Positive integers	To prevent a job for running into a workload's non-idle hours
Enabled	Whether the job will run	True/false	To disable a job without removing it
Full	For scheduling a full Garbage Collection job	Switch (true/false)	By default, every fourth job is a full Garbage Collection job. With this switch, you can schedule full Garbage Collection to run more frequently.
InputOutputThrottle	Specifies the amount of input/output throttling applied to the job	Integers 0-100 (indicates a percentage)	Throttling ensures that jobs don't interfere with other I/O-intensive processes.
Memory	The percentage of memory on the system that a job should use	Integers 0-100 (indicates a percentage)	To control what level of impact the job will have on the memory resources of the system
Name	The name of the scheduled job	String	A job must have a uniquely identifiable name.
ReadOnly	Indicates that the scrubbing job processes and reports on corruptions that it finds, but does not run any repair actions	Switch (true/false)	You want to manually restore files that sit on bad sections of the disk.
Start	Specifies the time a job should start	`System.DateTime`	The *date* part of the `System.Datetime` provided to *Start* is irrelevant (as long as it's in the past), but the *time* part specifies when the job should start.
StopWhenSystemBusy	Specifies whether Data Deduplication should stop if the system is busy	Switch (True/False)	This switch gives you the ability to control the behavior of Data Deduplication--this is especially important if you want to run Data Deduplication while your workload is not idle.

Modifying Data Deduplication volume-wide settings

Toggling volume settings

You can set the volume-wide default settings for Data Deduplication via the usage type that you select when you enable a deduplication for a volume. Data Deduplication includes cmdlets that make editing volume-wide settings easy:

- `Get-DedupVolume`
- `Set-DedupVolume`

The main reasons to modify the volume settings from the selected usage type are to improve read performance for specific files (such as multimedia or other file types that are already compressed) or to fine-tune Data Deduplication for better optimization for your specific workload. The following example shows how to modify the Data Deduplication volume settings for a workload that most closely resembles a general purpose file server workload, but uses large files that change frequently.

1. See the current volume settings for Cluster Shared Volume 1.

   ```
   Get-DedupVolume -Volume C:\ClusterStorage\Volume1 | Select *
   ```

2. Enable OptimizePartialFiles on Cluster Shared Volume 1 so that the MinimumFileAge policy applies to sections of the file rather than the whole file. This ensures that the majority of the file gets optimized even though sections of the file change regularly.

   ```
   Set-DedupVolume -Volume C:\ClusterStorage\Volume1 -OptimizePartialFiles
   ```

Available volume-wide settings

SETTING NAME	DEFINITION	ACCEPTED VALUES	WHY WOULD YOU WANT TO MODIFY THIS VALUE?
ChunkRedundancyThreshold	The number of times that a chunk is referenced before a chunk is duplicated into the hotspot section of the Chunk Store. The value of the hotspot section is that so-called "hot" chunks that are referenced frequently have multiple access paths to improve access time.	Positive integers	The main reason to modify this number is to increase the savings rate for volumes with high duplication. In general, the default value (100) is the recommended setting, and you shouldn't need to modify this.
ExcludeFileType	File types that are excluded from optimization	Array of file extensions	Some file types, particularly multimedia or files that are already compressed, do not benefit very much from being optimized. This setting allows you to configure which types are excluded.
ExcludeFolder	Specifies folder paths that should not be considered for optimization	Array of folder paths	If you want to improve performance or keep content in particular paths from being optimized, you can exclude certain paths on the volume from consideration for optimization.

SETTING NAME	DEFINITION	ACCEPTED VALUES	WHY WOULD YOU WANT TO MODIFY THIS VALUE?
InputOutputScale	Specifies the level of IO parallelization (IO queues) for Data Deduplication to use on a volume during a post-processing job	Positive integers ranging 1-36	The main reason to modify this value is to decrease the impact on the performance of a high IO workload by restricting the number of IO queues that Data Deduplication is allowed to use on a volume. Note that modifying this setting from the default may cause Data Deduplication's post-processing jobs to run slowly.
MinimumFileAgeDays	Number of days after the file is created before the file is considered to be in-policy for optimization.	Positive integers (inclusive of zero)	The **Default** and **Hyper-V** usage types set this value to 3 to maximize performance on hot or recently created files. You may want to modify this if you want Data Deduplication to be more aggressive or if you do not care about the extra latency associated with deduplication.
MinimumFileSize	Minimum file size that a file must have to be considered in-policy for optimization	Positive integers (bytes) greater than 32 KB	The main reason to change this value is to exclude small files that may have limited optimization value to conserve compute time.
NoCompress	Whether the chunks should be compressed before being put into the Chunk Store	True/False	Some types of files, particularly multimedia files and already compressed file types, may not compress well. This setting allows you to turn off compression for all files on the volume. This would be ideal if you are optimizing a dataset that has a lot of files that are already compressed.
NoCompressionFileType	File types whose chunks should not be compressed before going into the Chunk Store	Array of file extensions	Some types of files, particularly multimedia files and already compressed file types, may not compress well. This setting allows compression to be turned off for those files, saving CPU resources.

SETTING NAME	DEFINITION	ACCEPTED VALUES	WHY WOULD YOU WANT TO MODIFY THIS VALUE?
OptimizeInUseFiles	When enabled, files that have active handles against them will be considered as in-policy for optimization.	True/false	Enable this setting if your workload keeps files open for extended periods of time. If this setting is not enabled, a file would never get optimized if the workload has an open handle to it, even if it's only occasionally appending data at the end.
OptimizePartialFiles	When enabled, the MinimumFileAge value applies to segments of a file rather than to the whole file.	True/false	Enable this setting if your workload works with large, often edited files where most of the file content is untouched. If this setting is not enabled, these files would never get optimized because they keep getting changed, even though most of the file content is ready to be optimized.
Verify	When enabled, if the hash of a chunk matches a chunk we already have in our Chunk Store, the chunks are compared byte-by-byte to ensure they are identical.	True/false	This is an integrity feature that ensures that the hashing algorithm that compares chunks does not make a mistake by comparing two chunks of data that are actually different but have the same hash. In practice, it is extremely improbable that this would ever happen. Enabling the verification feature adds significant overhead to the optimization job.

Modifying Data Deduplication system-wide settings

Data Deduplication has additional system-wide settings that can be configured via the registry. These settings apply to all of the jobs and volumes that run on the system. Extra care must be given whenever editing the registry.

For example, you may want to disable full Garbage Collection. More information about why this may be useful for your scenario can be found in Frequently asked questions. To edit the registry with PowerShell:

- If Data Deduplication is running in a cluster:

```
  Set-ItemProperty -Path HKLM:\System\CurrentControlSet\Services\ddpsvc\Settings -Name DeepGCInterval
-Type DWord -Value 0xFFFFFFFF
  Set-ItemProperty -Path HKLM:\CLUSTER\Dedup -Name DeepGCInterval -Type DWord -Value 0xFFFFFFFF
```

- If Data Deduplication is not running in a cluster:

```
Set-ItemProperty -Path HKLM:\System\CurrentControlSet\Services\ddpsvc\Settings -Name DeepGCInterval
-Type DWord -Value 0xFFFFFFFF
```

Available system-wide settings

SETTING NAME	DEFINITION	ACCEPTED VALUES	WHY WOULD YOU WANT TO CHANGE THIS?
WlmMemoryOverPercentThreshold	This setting allows jobs to use more memory than Data Deduplication judges to actually be available. For example, a setting of 300 would mean that the job would have to use three times the assigned memory to get canceled.	Positive integers (a value of 300 means 300% or 3 times)	If you have another task that will stop if Data Deduplication takes more memory
DeepGCInterval	This setting configures the interval at which regular Garbage Collection jobs become full Garbage Collection jobs. A setting of n would mean that every n^{th} job was a full Garbage Collection job. Note that full Garbage Collection is always disabled (regardless of the registry value) for volumes with the Backup Usage Type. `Start-DedupJob -Type GarbageCollection -Full` may be used if full Garbage Collection is desired on a Backup volume.	Integers (-1 indicates disabled)	See this frequently asked question

Frequently asked questions

I changed a Data Deduplication setting, and now jobs are slow or don't finish, or my workload performance has decreased. Why? These settings give you a lot of power to control how Data Deduplication runs. Use them responsibly, and monitor performance.

I want to run a Data Deduplication job right now, but I don't want to create a new schedule--can I do this? Yes, all jobs can be run manually.

What is the difference between full and regular Garbage Collection? There are two types of Garbage Collection:

- *Regular Garbage Collection* uses a statistical algorithm to find large unreferenced chunks that meet a certain criteria (low in memory and IOPs). Regular Garbage Collection compacts a chunk store container only if a minimum percentage of the chunks is unreferenced. This type of Garbage Collection runs much faster and uses fewer resources than full Garbage Collection. The default schedule of the regular Garbage Collection job is to run once a week.
- *Full Garbage Collection* does a much more thorough job of finding unreferenced chunks and freeing more disk space. Full Garbage Collection compacts every container even if just a single chunk in the container is unreferenced. Full Garbage Collection will also free space that may have been in use if there was a crash or

power failure during an Optimization job. Full Garbage Collection jobs will recover 100 percent of the available space that can be recovered on a deduplicated volume at the cost of requiring more time and system resources compared to a regular Garbage Collection job. The full Garbage Collection job will typically find and release up to 5 percent more of the unreferenced data than a regular Garbage Collection job. The default schedule of the full Garbage Collection job is to run every fourth time Garbage Collection is scheduled.

Why would I want to disable full Garbage Collection?

- Garbage Collection could adversely affect the volume's lifetime shadow copies and the size of incremental backup. High churn or I/O-intensive workloads may see a degradation in performance by full Garbage Collection jobs.
- You can manually run a full Garbage Collection job from PowerShell to clean up leaks if you know your system crashed.

Additional performance tuning resources

12/16/2022 • 2 minutes to read • Edit Online

Use the links in this topic to learn more about the concepts that were discussed in this tuning guide.

Microsoft Windows Server Websites

- Windows Server Catalog
- Windows Sysinternals
- Transaction Processing Performance Council
- Windows Assessment and Deployment Kit

Power Management Tuning Resources

- Power Policy Configuration and Deployment in Windows
- Using PowerCfg to Evaluate System Energy Efficiency
- Interrupt-Affinity

Networking Subsystem Tuning Resources

- Scalable Networking: Eliminating the Receive Processing Bottleneck—Introducing RSS
- Windows Filtering Platform
- Networking Deployment Guide: Deploying High-Speed Networking Features

Storage Subsystem Tuning Resources

- Disk Subsystem Performance Analysis for Windows (Parts of this document are out of date, but many of the general observations and guidelines captured are still accurate and relevant.)

File Server Tuning Resources

- Performance Tuning Guidelines for Microsoft Services for Network File System
- [MS-FSSO]: File Access Services System Overview
- How to disable the TCP autotuning diagnostic tool

Active Directory Server Tuning Resources

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

Virtualization Server Tuning Resources

- What's New in Hyper-V in Windows Server 2016

- Hyper-V Dynamic Memory Configuration Guide
- NUMA Node Balancing
- Hyper-V WMI Provider
- Hyper-V WMI Classes
- About Virtual Machines and Guest Operating Systems
- Optimizing and Troubleshooting Hyper-V Storage
- Optimizing and Troubleshooting Hyper-V Networking

Print Server Tuning Resources

- Print Server Scalability and Capacity Planning

Server Workload Tuning Resources

> **NOTE**
> Microsoft Server Performance Advisor is no longer available. Please use the resources in this section instead.

- Performance Tuning for NTttcp
- Ttcp
- How to use NTttcp to Test Network Performance
- Using the File Server Capacity Tool
- Using the SPECsfs2008 File Server
- Performance Tuning for the Sales and Distribution Workload
- Performance Tuning for Online Transaction Processing (OLTP)
- How to: Configure SQL Server to Use Soft-NUMA
- How to: Map TCP/IP Ports to NUMA Nodes
- ALTER SERVER CONFIGURATION (Transact-SQL)

Performance Tuning Guidelines for previous versions of Windows Server

Use the performance tuning guidelines to improve performance for older versions of Windows Server.

Here's a list of performance tuning guidelines for previous versions of Windows Server:

- Performance Tuning Guidelines for Windows Server 2016
- Performance Tuning Guidelines for Windows Server 2012 R2
- Performance Tuning Guidelines for Windows Server 2012
- Performance Tuning Guidelines for Windows Server 2008 R2
- Performance Tuning Guidelines for Windows Server 2016