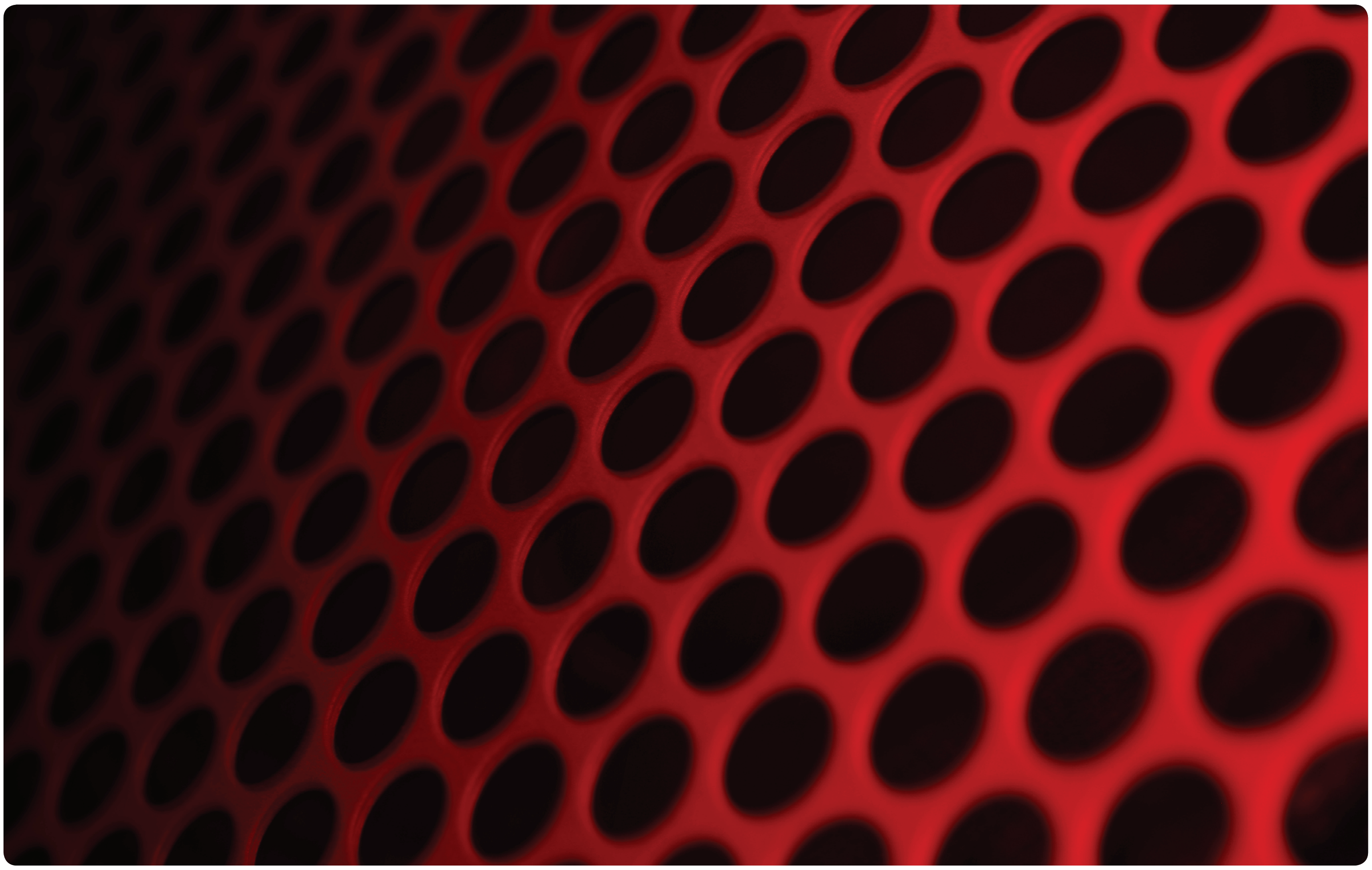
Memory and SSD Optimization

In Windows Server 2012 and SQL Server 2012



Contents

Windows Server 2012 Overview

Windows Server 2012 represents the most advanced server operating system that Microsoft has ever released. Virtually every aspect of the OS has been enhanced, including Active Directory, IIS and File Services. However, the newest, most exciting changes are advancements to Hyper-V.

Microsoft entered the mainstream server virtualization market in 2008 with the first release of Hyper-V in Windows Server 2008. It was their first attempt at a type-1 Hypervisor, and while it was met with optimism, its overall capabilities fell short of expectations.

In 2010, Microsoft released Windows Server 2008 R2, which included an updated Hyper-V component. Hyper-V R2, and the subsequent release of Service Pack 1 added many capabilities and features that brought the solution closer to being competitive with VMware's platform.

With the release of Windows Server 2012, Microsoft has closed the gap on VMware vSphere's capacity and feature set, and has added some functionality that is missing from VMware vSphere 5. Below is a comparison of capacity differences in Hyper-V R2 vs. Server 2012 Hyper-V vs. VMware vSphere 5.

	Windows Server 2008 R2 Hyper-V	Windows Server 2012 Hyper-V	VMware vSphere 5.0
Maximum Hyper-V Host Logical Processors	64	320	160
Physical Memory Addressed by Hypervisor	1 TB	4 TB	2 TB
Maximum Virtual CPUs per Host	512	2,048	2,048
Virtual CPUs per Virtual Machine	4	64	32
Memory per Virtual Machine	64 GB	1 TB	1 TB
Total Active VMs per Host	384	1,024	512
Maximum Hosts per Cluster	16	64	32
Maximum VMs per Cluster	1,000	4,000	3,000

Table 1, Capacity differences in Hyper-V R2, Hyper-V 2012 and VMware vSphere 5

The table above shows that Server 2012 Hyper-V is capable of driving large virtualization loads, and in particular, is able to manage significantly more memory and more virtual machines than previous iterations of Hyper-V. Therefore, proper optimization of memory and storage configurations is more critical than ever to achieve optimal performance.

Memory and Storage Optimization in Hyper-V

Memory is most commonly the #1 constraint in virtualization density; CPU speeds and core counts have continued to escalate, but memory remains the limiting factor in the number of virtual machines running per virtualization host, whether that platform is based on VMware or Microsoft.

Both companies have memory management capabilities included in their Hypervisor, and both use a different approach to optimize the use of memory in the system. VMware allocates the configured memory to virtual machines at power on and then attempts to recover memory later using the techniques below. Hyper-V, however, takes the opposite approach and grants systems a smaller amount of RAM at start up, and then dynamically adds more memory as running VMs require it. Both techniques accomplish the same goal: Optimize the use of available memory while providing VMs what they need to run effectively.

Understanding NUMA Impact

CPUs run much faster than the memory attached to the system. Legacy computer designs only allowed a single CPU to access the memory bus at any single time, so as CPU and core counts increased, performance was stunted because processes running on different processors couldn't access memory simultaneously.

Non-Uniform Memory Access (NUMA) is a memory architecture used in modern servers and addresses this shortcoming by breaking up CPUs and system memory into nodes, where each CPU socket is attached to a unique memory controller and related DIMMs. CPUs are able to access their local NUMA node memory faster than non-local memory in another node. All nodes are interconnected so that processes in one node may access memory in a remote node, albeit with a performance penalty. This is referred to as NUMA spanning.

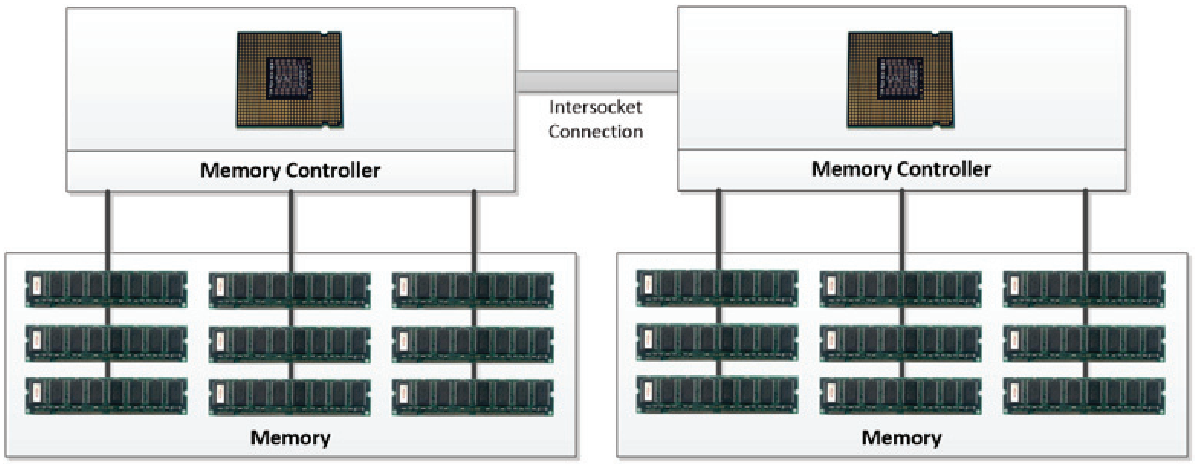


Figure 1, NUMA node and memory

Newer operating systems, including Windows Server 2012, are NUMA-aware and will automatically allocate memory to processes from the local NUMA node to optimize performance. However, NUMA can have a significant impact on virtualization; virtual machines are simply running processes on the Hypervisor, and can use a significant amount of memory.

The NUMA boundary is calculated by dividing the memory in the server by the number of CPU sockets in the system. According to Microsoft , performance is reduced by almost 8% when the amount of memory allocated to a virtual machine is larger than the NUMA boundary.

By default, Hyper-V assigns the NUMA node preference every time the VM is started. When selecting a NUMA node, Hyper-V will attempt to allocate a node that has enough memory to support the entire running virtual machine. Virtual machines run optimally when the virtual CPUs and the VM's memory on the same NUMA node.

For best performance, ensure that virtual machines on a Hyper-V host are allocated the correct amount of memory in relation to the NUMA boundary. Lower memory capacity and core-count hosts may have challenges running higher memory capacity VMs.

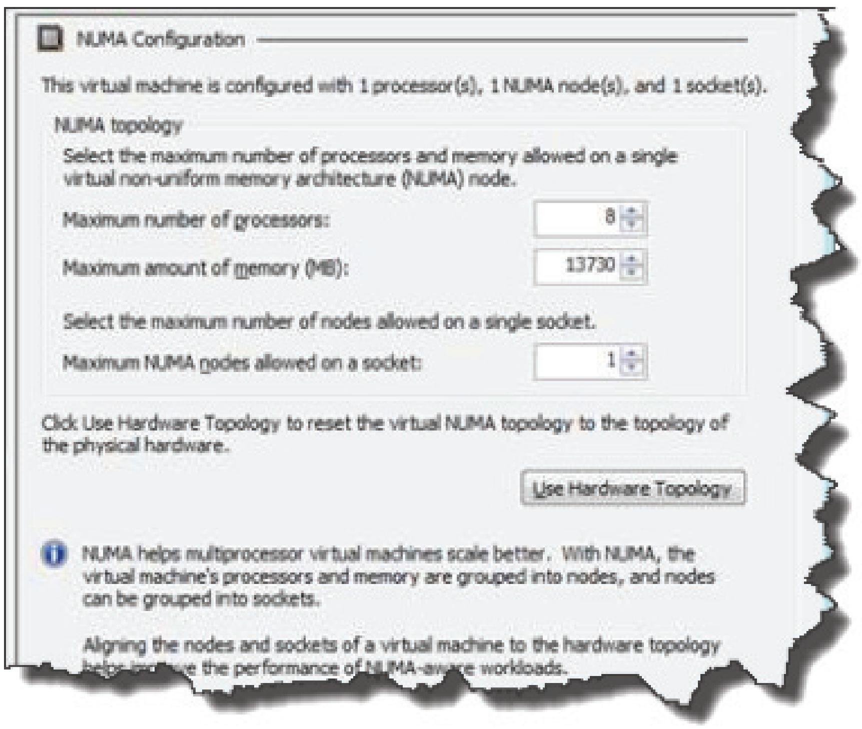
Finally, Hyper-V hosts also have a preference to enable or disable NUMA spanning. Spanning is enabled by default so that the most memory can be allocated to all running virtual machines, even if it means VM memory may span nodes. To optimize performance for sensitive workloads, spanning may be disabled in the Hyper-V settings of the host.

Virtual NUMA Support in Hyper-V 2012

Hyper-V 2012 introduces the concept of NUMA for virtual machines. A VM's virtual CPUs and memory are grouped into virtual NUMA nodes based on the underlying physical topology of compute and memory resources.

Hyper-V leverages a standard ACPI model for presenting this topology so that any NUMA-aware operating system, including Windows and Linux, can leverage virtual NUMA. Virtual NUMA means that guest operating systems are able to self-optimize NUMA scheduling based on their configuration alignment to the host's hardware NUMA architecture.

Virtual NUMA is configured automatically in a virtual machine based on the underlying hardware configuration on the system where the VM was created. However, this information can be customized in the interface for specific needs.


NUMA Configuration

This virtual machine is configured with 1 processor(s), 1 NUMA node(s), and 1 socket(s).

NUMA topology

Select the maximum number of processors and memory allowed on a single virtual non-uniform memory architecture (NUMA) node.

Maximum number of processors: 8

Maximum amount of memory (MB): 13730

Select the maximum number of nodes allowed on a single socket.

Maximum NUMA nodes allowed on a socket: 1

Click Use Hardware Topology to reset the virtual NUMA topology to the topology of the physical hardware.

NUMA helps multiprocessor virtual machines scale better. With NUMA, the virtual machine's processors and memory are grouped into nodes, and nodes can be grouped into sockets.

Aligning the nodes and sockets of a virtual machine to the hardware topology helps improve the performance of NUMA-aware workloads.

Figure 2, Virtual NUMA configuration in a VM

Finally, Hyper-V 2012 contains specific controls when VMs need to be migrated between hosts with dissimilar NUMA topologies.

While Virtual NUMA is a great addition to Hyper-V, it only exacerbates the need to ensure the proper amount of memory and the placement of memory in the physical hosts is optimized. Since virtual machines receive their default virtual NUMA configuration from the underlying hardware, incorrect NUMA optimization will result in less-than-optimal performance.

Hyper-V Dynamic Memory

While VMware vSphere allocates all memory to VMs at startup and then attempts to reclaim unused memory though transparent page sharing and ballooning, Hyper-V's Dynamic Memory takes a different approach to managing memory. Hyper-V does not swap any VM memory to disk, so in earlier versions of Hyper-V, memory configured for a virtual machine was 100% allocated at startup. This caused memory to be a considerable bottleneck in Hyper-V virtual machine density.

Dynamic Memory is a recent new feature of Hyper-V and aims to automatically right size virtual machine memory. Dynamic Memory treats all memory in the host system as a single pool and allocates memory to running virtual machines automatically based on demand.

Rather than allocate all of the VM memory at startup, Hyper-V provides configuration options to set different parameters for startup RAM, minimum RAM and maximum RAM. This allows administrators to tune the memory characteristics of the virtual machines and how they operate in Hyper-V.

Startup Memory	The amount of memory presented to the guest operating system when the VM is started on the Hyper-V host
Minimum Memory	The minimum amount of physical memory that the VM must be using; may be lower than Startup Memory
Maximum Memory	The maximum amount of physical memory that the VM may be allocated by Dynamic Memory
Memory Buffer	The amount of memory headroom between the amount of memory granted and the amount of memory currently needed by the guest operating system
Memory Weight	The importance of memory for a virtual machine compared to other VMs on the host; VMs with higher weight are given preferential access to physical RAM

Dynamic Memory in Action

When a virtual machine is powered on, the Startup Memory is immediately allocated to the virtual machine. Once the Hyper-V Integration Services are loaded, the hypervisor begins evaluating the memory demand of the guest OS. The Memory Buffer setting is added to the current memory demand to determine if there is too much or too little RAM currently allocated to the VM. Once the memory demand rises to the point where it is inside of the buffer cushion, Hyper-V will increase the allocation, adding memory to VM hardware configuration.

For instance, for a VM with a Startup Memory setting of 1 GB and a Memory Buffer setting of 20% (default), Hyper-V will increase the memory allocation for that VM once the demand rises above 854 MB (855MB + 20% = 1026MB).

With Windows Server 2012 Hyper-V and Dynamic Memory, virtual machines are able to start up with less memory than they may need in typical operation, and Dynamic Memory will add RAM to the VM as required. Likewise, it's feasible to configure a VM with more Startup Memory, and then allow the VM to reduce the amount of memory used after the guest operating system is running. The Memory Buffer setting indicates how much "headroom" is necessary between the current allocation and the current demand.

Dynamic Memory allows a Hyper-V host to start and run more virtual machines than otherwise possible with the total physical memory in the host. This also allows memory to be smartly allocated to virtual machines based on just-in-time demand. If there is less physical memory in the host than current demand, Hyper-V employs a form of ballooning in which a component in the Hyper-V guest OS integration services coordinates with the host to analyze and free unused memory and return it to the host for reallocation to other VMs.

Dynamic Memory does not work in all circumstances. Applications that perform their own memory management, such as SQL Server, may not make good candidates for Dynamic Memory. These applications continue to consume memory as cache space up to the maximum available, resulting in VMs that are always running at their configuration maximum. At the very least, consider making application adjustments to prevent these applications from artificially inflating memory demand.

Dynamic Memory may also adversely affect applications where memory allocation occurs only at startup. In these cases, the VM may not run at an optimum performance level as the true nature of memory demand may not be exposed to Dynamic Memory. In these cases, VMs will need to be configured with a higher Startup RAM value and a lower Minimum RAM value. This will allow the VM to start with a higher amount of memory, but then reduce memory consumption when the Dynamic Memory is active and recognizes the VM does not need that amount of RAM.

While this is great functionality to right-size virtual machine memory use, it should not be confused with over-commitment of memory. Hyper-V Dynamic Memory provides memory oversubscription, not over-commitment; the latter implies there is not enough memory to properly run the workload. Dynamic Memory is not a substitute for good, sound memory management. Administrators must have an understanding of the memory requirements of virtualized systems. Performance Monitor counters and System Center Operations Manager are excellent ways to gain knowledge on the true memory needs of systems in Hyper-V.

Storage Optimization in Hyper-V

Storage Spaces

Windows Server 2012 offers a new storage subsystem called Storage Spaces, in which internal and external drives can be configured as storage pools. Storage Spaces can be configured from these pools and leverage features like thin-provisioning, disk hot-add and RAID.

Hyper-V Smart Paging

Windows Server 2012 introduces a new feature called Smart Paging, which is intended to improve the reliability of virtual machine restart operations when Dynamic Memory is used. When virtual machines are configured with a lower Minimum value than the Startup value, it's possible that in a restart situation, the VM may not start if ample free RAM is not available for the VM to meet the Startup RAM requirement. To mitigate this, Server 2012 can leverage Smart Paging to make up the difference between Minimum RAM and Startup RAM. Smart Paging is used to bridge the gap between Minimum memory and Startup memory.

Smart Paging creates a page file on the Hyper-V host storage equal in size to the difference of the VM's Minimum and Startup RAM. When the VM is restarted and there is not enough memory to allocate the Startup RAM value, a pagefile will be used to create virtual memory to make up the difference. The pagefile is only created if Smart Paging is needed to restart the VM and is only used until the Dynamic Memory driver loads in the VM; at that point the VM memory allocation is reduced to the Minimum RAM number and the pagefile is deleted. To minimize performance impact, Smart Paging is only used in the following circumstances:

- A virtual machine is restarted
- There is no available physical memory in the host
- No memory can be reclaimed from other running virtual machines

Smart Paging is not used when a VM is being administratively started from a "stopped" state. It is only used when VMs are restarted. It is also not used when failing VMs over in a Hyper-V cluster when a host failure occurs. Smart Paging is only intended to prevent the case where a VM is restarted and memory is constrained. The VM that was running is able to return to a running state.

To optimize performance, use SSD storage, such as Kingston's Enterprise Grade SSDNow products, for placement of the Smart Paging files. This ensures that in the event of a memory-starved system, VM restart operations happen as fast as possible.

Guest-Paging Optimization

Operating systems leverage virtual memory to extend the amount of available memory in the system. Paging is never optimal, but in some cases it may be unavoidable. Hyper-V Dynamic Memory operates on the principle of guest paging. Hyper-V does not swap VM memory to disk; rather is puts pressure on the guest operating system through ballooning, which in turn allows the guest to best decide which contents in memory should be sent to the Windows paging file.

For systems where memory may fall under contention, breaking guest paging files to a different VHD that resides on SSD will significantly improve performance. For high-performance critical workloads, leveraging SSD for guest paging may overcome temporary issues when physical host memory becomes scarce.

Optimizing VDI Workloads

VDI is a challenging solution to any organization. On a smaller scale, it is easily managed, but when grown to larger implementations, the challenges of memory and I/O management become paramount. VDI is a very different solution than server virtualization. The lessons learned when virtualizing a datacenter full of servers do not translate to a successful VDI plan. The I/O characteristics of VDI are a far cry from server workloads.

User experience is the number one determining factor in the success of a VDI implementation. Users expect the performance of a VDI solution to be at least equal, if not better, than their traditional desktop experience. Therefore, ensuring good performance is critical.

Memory and VDI

The primary purpose of a desktop is to provide a means for a user to gain access to and use applications, so applications are the determining factor in memory consumption of the desktop.

Dynamic Memory will increase the consolidation ratio of VDI workstations on a single server because the fluctuating memory needs of one virtual desktop to the next will allow some systems to use more memory that others and will automatically allocate the RAM required. Consider the following example of a Hyper-V host with 96GB RAM in both a static memory and Dynamic Memory configuration.

	Static Memory	Dynamic Memory
Host Memory Capacity	96GB	96GB
VM Memory Configuration	2 GB (Static)	1 GB Startup, 2 GB Maximum
Average Memory Usage	1.2 GB	1.2 GB
Total VMs per Host	47	78

Table 2, Dynamic Memory effect on VDI density

In the static memory example, the highest number of VMs possible on the host is about 46 or 47 VMs (47 x 2GB = 94GB). In the Dynamic Memory example, the same configuration and workload can support up to 78 VMs since only the memory required for each VM is allocated to it (78 x 1.2GB = 94GB). Dynamic Memory with Hyper-V in VDI scenarios can increase the number of VMs per host significantly.

Storage and VDI

While memory will improve density for VDI, storage I/O performance is the number one technical reason that VDI fails. Traditional desktop storage consists of a single consumer-grade hard drive responsible for a single operating system and user. While the performance of these mechanical drives is relatively poor, users don't see a problem because drives are able to keep up with the demands of one user. However, when multiplied by thousands of users, the I/O demand for simple things like logging in to the VDI workstation ("bootstorm") becomes a major issue.

Persistent vs. Non-Persistent VDI

Persistent VDI is a virtual desktop solution where the state of the virtual machine acts like any other traditional desktop and the changes made to the system survive logoff of the end user. Persistent VDI desktops are "assigned" to an end user so that they are always using the same virtual desktop image each time they connect.

Persistent VDI is a one-to-one desktop image; each image is a unique desktop and consumes approximately 20GB – 40GB of storage. For a VDI deployment of 1000 persistent desktops, the storage demand is about 30TB. Not only is the storage cost extremely high, but the IOPs demand on the storage is the same as non-persistent.

In non-persistent, or stateless VDI, end users access a shared image that is used to create many identical desktops. User changes to the desktop are not saved, and user personality information and data locations are redirected to central storage locations. Stateless VDI saves significant amounts of SAN-based storage since hundreds of desktops are created using a single desktop image or collection of images, and only the uniqueness of each (the "delta") is temporarily stored while the user is connected. Upon user logoff, the desktop uniqueness storage is flushed and the virtual desktop returns to a previous, pristine state.

Non-persistent VDI is a one-to-many desktop image. Each desktop is sourced from a single read-only image that contains the OS, and then a small 2GB-4GB "delta" disk that contains the uniqueness for that system. This results in significantly less storage use; the same 1000 desktop environment would require only the initial 20GB – 40GB for the read only image, and then approximately 2TB – 4TB for delta disks.

Leveraging SSD for VDI

Non-persistent VDI is the panacea of VDI. It is also the most challenging to attain because of the applications readiness necessary and the knowledge of the user required to construct a desktop on-demand. However, non-persistent VDI holds the key to a sound ROI model because of the significantly reduced storage.

The I/O demand on VDI is the same regardless of persistent or non-persistent models. However, with non-persistent VDI, the read IOPs are split from the write IOPs onto different disk targets, which allows for the use of SSD for either read or write IOPs, or even both.

Windows 7 VM generates and average of about 230 read/write IOPs for the boot and login process. Most of that is read traffic during the boot process, but an average of 13 IOPs is seen during login. Remember, in VDI, user experience is everything. If 1,000 people log in at the same time, the storage system behind the VDI solution could see an average of 230,000 IOPs at any given moment, with over 13,000 of those being write IOPs. [2]

SSDs are capable of over 70,000 IOPs per drive, and when aggregated into arrays, can easily top hundreds of thousands of IOPs. And because of the reduced storage requirements in non-persistent VDI, SSD offers the best performance per-GB of any storage medium.

[2] Information on IOPs load was obtained from Project VRC testing results. See http://www.projectvrc.com/white-papers for more information.

Optimizing Memory and I/O with SQL Server 2012

SQL Server is a key application in the data center that demands precise storage architecture and memory optimization. SQL Server 2012, as with previous versions, depends heavily on a properly designed back-end architecture.

Memory Behavior in SQL Server

By default, SQL Server will automatically adjust memory usage based on the available physical memory and the other running processes in the operating system. In a virtual world, this translates to the amount of memory configured for the virtual machine.

SQL Server memory behavior is controlled by the **min server memory** and **max server memory** settings. By default, **max server memory** on a 64-bit server (the maximum setting) defaults to the total amount of RAM available in the system. If the core SQL Server service is the only service running on the system (aside from typical OS services), SQL Server will eventually consume the majority of RAM available. Other SQL Server services, such as SQL Server Integration Services (SSIS) or SQL Server Reporting Services (SSRS), will consume some of the RAM available to the core SQL Server service, but eventually Windows Task Managerwill report all memory is being used by the system.

This issue is compounded when more than one SQL instance is present on the system. Each SQL instance requires its own pool of memory, and when multiple SQL instances are running on the same system, the instances will begin to compete for resources. In the default SQL configuration, each instance will attempt to consume as much memory as possible, with neither getting the proper resources they need to perform optimally.

In a virtual world, VMs can easily be configured with any amount of RAM desired, so this behavior has adverse effects on virtual machines that leverage Dynamic Memory.   Hyper-V uses the memory demand + buffer size to determine the necessary memory allocation for the VM. As demand increases, Hyper-V looks at the current allocation and the buffer setting to determine if the VM needs its memory increased. As more memory is added to the VM, SQL Server sees there's free memory and then consumes additional buffer pool space, which in turn causes Hyper-V to

Changes to Memory Management in SQL 2012

*In previous versions of SQL Server, the **max server memory** configuration setting only affected the SQL buffer pool (Bpool) size, which only affects the memory allocated to the buffer pool for cache; it doesn't affect the amount of memory that SQL Server can use as a whole. CLR and Multipage allocations would occur outside of the buffer pool and cause SQL to use more memory than expected.*

*SQL Server 2012 contains a redesigned Memory Manager that plays a more poignant role in managing SQL memory-consuming components beyond the buffer pool, so that **max server memory** now affects all allocations for SQL. This makes sizing SQL memory consumption more predictable.*

re-evaluate the memory buffer and increase the allocation. This stepping effect will lead to the eventual consumption of memory allocation up to the Hyper-V Maximum Memory setting for the VM.

The answer is either to use a static memory configuration for the virtual machine, configure the SQL **max server memory** parameter to control memory use, or both.

Determining Memory Requirements for SQL Server

In most production implementations of SQL Server, disable automatic memory allocation and configure the desired memory usage. This approach will ensure each instance is contained within a finite memory space. To determine the proper memory configuration, each instance will need to be run while reviewing key performance counters over time. Following performance counters will reveal the required memory under different conditions.

Start with a baseline amount of memory in the system to provide adequate performance while monitoring use. Then, monitor the counters in Table 3 and Table 4. The following are system-level memory counters that can be used to monitor for a low memory condition.

Performance Counter	Measurement	Guidance
Memory → Available Bytes	The number of bytes of memory are currently available for use by processes	SQL attempts to maintain from 4-10MB of free physical memory. The remaining physical RAM is used by the operating system and SQL Server Low Available Bytes may indicate that max server memory is not configured
Memory → Pages/sec	The number of pages that either were retrieved from disk due to hard page faults or written to disk to free space in the working set due to page faults	Measures the number of pages per second that are paged out from RAM to disk. Higher the value, higher will be I/O activities and will result in decrease in performance. If you have only SQL server application running on the server then in most cases this value should be near zero. However you don't see much performance degradation until it is 20, when SQL Server is not the only application. Above 20, it is an indication to have more RAM on the server

Table 3, Windows memory performance counters

In addition, the following Performance Monitor counters are used to monitor the amount of memory being used by SQL Server:

Performance Counter	Measurement	Guidance
Process → Working Set	The amount of memory used by a specific process (SQL)	If consistently below the amount of memory configured for the SQL instance(s) (*min server memory* and *max server memory*), SQL is configured for more memory than required.

SQL Server → Buffer Manager → Buffer Cache Hit Ratio	Indicates the percentage of pages found in the buffer cache without having to read from disk	A ratio of 90 percent or higher is optimal. If below, add memory until this value is consistently greater than 90 percent
SQL Server → Memory Manager → Total Server Memory (KB)	The amount of memory SQL has committed using the Memory Manager	If the Total Server Memory (KB) counter is consistently high compared to the amount of physical memory in the system, it could indicate that more memory is required.
SQL Server → Memory Manager → Free Memory (KB)	The amount of committed memory currently not used by SQL	Indicative of too much memory in the system based on the workload; however, without configuring *max server memory*, this will likely be unreliable

Table 4, SQL Server memory performance counters

Use these counters to determine the proper values for min server memory and max server memory on each SQL instance.

Optimizing Storage Performance in SQL Server

While memory will affect SQL performance, a slow disk subsystem will create issues just as easily. Given SQL Server's heavy storage I/O requirements, storage environments must be carefully designed to ensure sustained performance.

The evolution of SSDs has become a game-changer for SQL performance; read and write performance of a few SSD drives rival that of large-scale SAN environments with high numbers of spindles. When architecting storage infrastructures supporting SQL Server, there are a variety of areas that come into play:

- Placement of the databases and the transaction logs
- Placement of the Tempdb

SSDs present unparalleled read and write performance characteristics. Each database has specific characteristics that will determine if SSD is a good fit. Static databases that store large amounts of historical data may not be good candidates for SSD since there won't be high I/O demands. The cost per GB of SSD is higher than mechanical storage, and historical data that does not require frequent access will not benefit from SSD. The following sections provide recommendations for leveraging SSDs with various SQL components.

Note: Not all SSDs are ideal for server use. Look for SSDs such as Kingston's Enterprise line of SSDs are designed with premium NAND components and optimized SSD controllers Server environments that require higher durability and reliability.

User Databases

User database I/O performance requirements will vary on a case-by-case basis. Some databases are very write intensive while others are more read-oriented. A properly design SQL database will include the use of indexes that speed common lookups, and the allocated buffer space in memory should handle most read operations. Therefore, user databases can be stored on traditional mechanical storage in the necessary RAID configuration for performance and redundancy.

Transaction Logs

Transaction logs use sequential writes in operation. Performance of transaction logs is paramount for fast database operation, so SSD is a perfect solution. Transaction logs require redundancy, so SSDs should be arranged in a RAID array to provide resiliency.

In situations where multiple databases are hosted on a centralized SQL Server or instance, the best practice has historically been to locate each database's transaction log on a separate array or disk to optimize their sequential I/O characteristics; combining them would essentially turn the collection of transaction logs into a large random-I/O scenario, which is less than optimal for mechanical drives.

The performance of SSD can really be exploited by leveraging a single SSD array for all transaction logs since SSDs excel at random I/O.

TempDB

The Tempdb system database is a global resource available to all users connected to an instance of SQL Server. Tempdb is used for the following:

- Temporary user objects, such as: temporary tables, temporary stored procedures, or table variables
- Internal objects created by the SQL database engine (working tables to store intermediate results/sorting)
- Some instances of row versions that are generated by data modification transactions

Tempdb is subject to high-I/O, but is only temporary holding space. No real database data is permanently stored here and backups are not permitted. Each time the SQL instance is started, a fresh version of tempdb is created.

While tempdb doesn't require high resiliency, it is subject to performance issues if the underlying storage can't keep up with the workload. Tempdb is subject to a high write I/O, and its relatively low requirement for resiliency makes it a perfect candidate for SSD storage.

Virtualizing SQL Server

While Microsoft fully supports virtualizing SQL, many administrators have held off due to lack of confidence in sustained performance in a virtual state. As discussed in the beginning of this paper, Hyper-V is capable of running extremely large virtual machines. From a capacity perspective, Hyper-V 2012 is more than capable of running even the largest SQL servers. The key is in the level of oversubscription and guaranteeing that the resources required to effectively run a SQL server are available when the system needs it.

Understanding NUMA architecture is important to designing a powerful SQL platform (see page 4). In Hyper-V 2012, guest operating systems are NUMA-aware. Since SQL Servers often require large amounts of memory, ensure that the VM hardware will properly align to the NUMA architecture of the underlying hosts. For example, if a Hyper-V host only contains 96GB RAM, creating a 64GB SQL VM will result in immediate NUMA spanning since the most memory allocated to a single NUMA node is 48GB. To prevent NUMA spanning altogether, disable it in the Hyper-V host for critical SQL VMs with large amounts of memory.

Lastly, be cautious of oversubscribing a Hyper-V host that is running SQL VMs. The allure of server virtualization is the ability to consolidate multiple physical servers into one, and it isn't uncommon to see consolidation ratios of 20:1. However, server consolidation is about creating efficiency in leveraging idle time to run other workloads, so for a system that only requires 5% of a CPU is idle the other 95% of the time, during which time Hyper-V can run other VMs.

If a SQL Server requires 50% of a physical server's CPUs to run effectively, then it will require the same CPU utilization in a virtual state. That means consolidation ratios will be lower on Hyper-V hosts that run high-utilization SQL VMs. When virtualizing SQL, do not oversubscribe CPU capabilities of the Hyper-V hosts, and leverage Hyper-V's ability to adjust the CPU and memory priority so that the SQL VMs receive preferential access to those resources.

Summary

Memory and storage are two key areas of impact with regards to virtualization and SQL Server workloads. Windows Server 2012 brings a plethora of new capabilities and performance improvements that make optimization critical.

Use the following guidance discussed in this paper for memory optimization.

- For physical server provisioning, always ensure memory is populated properly in the memory banks following triple-channel or quad-channel memory architecture and NUMA. Visit http://www.kingston.com/us/business/server_solutions for more information on optimizing server configuration for performance.

- When leveraging Hyper-V, use Dynamic Memory to ensure the investment in memory is used most effectively. However, use care with workloads that may have memory configuration settings that are not configured optimally, such as database applications – these could lead to unpredictable results with Dynamic Memory use.

Always understand the true memory and storage performance requirements of any workload to be virtualized.

- Consider placing guest paging files on separate VHDs housed on SSDs, particularly for critical workloads. This will ensure that if Hyper-V host memory becomes constrained, resulting in-guest paging at the best possible performance. Also consider using SSDs for Hyper-V Smart Paging files to optimize performance in memory constrained situations.

- When virtualizing SQL, do not oversubscribe CPU and memory on the server. Analyze the SQL Server instance(s) to determine the memory requirements and then configure the max server memory setting to prevent unnecessary consumption of memory for the buffer pool.

Hyper-V Dynamic Memory may or may not benefit SQL Server. If the instances are properly configured for max server memory, then the amount of memory required for the VM is predictable and Dynamic Memory may not be required.

If the max server memory setting will not be configured, use static memory in the VM configuration; do not use Dynamic Memory, and only place one SQL instance on a server.

- Use server-grade SSDs (such as Kingston E100) for specific SQL databases or transaction logs. The SQL Tempdb will benefit from SSDs' performance and can easily be leveraged in a local server to avoid steep SAN costs.

Rather than placing each database's transaction logs on a different storage volume, consolidate them onto a single SSD array to benefit from the increased performance.

- Non-persistent VDI storage performance will benefit significantly from placing differencing (delta) disks on SSD storage. And while challenging, non-persistent VDI is the target model for a successful ROI on VDI solutions.

About the Authors

Michael Burke is the CTO of VDX, a leading professional services company headquartered in the Northeast United States focusing on desktop transformation and private cloud. Burke has over 10 years' experience working closely with Microsoft products. He has written many technical articles and whitepapers around Microsoft products and technologies and has been a guest speaker at several international conferences on the subject.

William Gray is a Solutions Architect at VDX. William has over 7 years' experience working with Microsoft products. He has worked with clients in a variety of industries building comprehensive, next generation solutions with Microsoft technologies.