Tintri

IntelliFlash™ Best Known Methods

Microsoft® SQL Server®

Table of Contents

Overview

Microsoft SQL Server is a complete database engine, for both relational and analytical data, and provides an enterprise platform to store and retrieve data for mission-critical business applications. While the majority of the working set of data resides in memory in SQL Server, the storage subsystem provides the platform to store the data so that it persists between service restarts, and so that massive amounts of data can be stored without requiring the same amount of system memory.

This technical document provides best known methods from Tintri for deploying and managing Microsoft SQL Server databases on Tintri's IntelliFlash storage systems. The guidance covers all IntelliFlash storage systems.

This guide is intended for both experienced SQL Server and infrastructure administrator professionals who are familiar with common SQL Server administration and management procedures. This guide is not a step-by-step tutorial but is designed to be comprehensive in nature. Tintri can provide experienced SQL Server professionals in conjunction with its consulting partners.

This best practice guide for SQL Server is not limited to the Microsoft Windows Server operating system, as SQL Server is now supported on the Linux operating system since the release of SQL Server 2017. Many of these best practices encompass recommendations for operating systems from Windows Server® 2008 to 2019, Linux®, virtualization platforms including Microsoft Hyper-V® and VMware vSphere®, and SQL Server 2008 to 2019 (currently in CTP at the time of this writing).

Introducing IntelliFlash Storage Systems and SQL Server Storage

Tintri's IntelliFlash storage systems feature a combination of powerful CPUs, DRAM, and SAS or NVMe™ SSDs to create a pool of storage resources. IntelliFlash systems each possess two active storage controllers, capable of maximizing performance while providing high availability with instantaneous failover. For ultimate performance, consider deploying IntelliFlash N-Series NVMe-Flash systems. The N-Series leverages the speed and efficiency of NVMe to deliver maximum performance to the most demanding enterprise applications.

The SQL Server engine itself is actually made up of two components: the storage engine, which manages the retrieval and placement of data on operational storage; and the query engine, which translates a program's database request, instructs the storage engine to retrieve the data, processes the data with the operation dictated by the program using SQL Server buffer pool memory to store this retrieved data, and then delivers the data to the program.

To SQL Server, the storage is arguably the most important part of the system stack. Data must be hardened on media for any write operation, and the faster the storage subsystem, the faster the write operation completes. Any data that is being read for fulfilling a request that is not already in SQL Server's reserved area of system memory, called the buffer pool, is retrieved from media as needed, which is now dependent on the speed of the storage. Suboptimal code or lack of memory in the buffer pool can cause excessive storage demand. The use of In-Memory OLTP for persisted data is reliant on the speed of storage for any data that changes, which can slow the operation down by orders of magnitude. The speed at which these processes can consume significant amounts of I/O is quite high, and the slower the storage system, the slower these operations.

To protect against system slowdown, ensure that your storage is as fast as your budget allows, but also performing to the best of its abilities. Not evaluating the system components between the SAN and database can almost guarantee that an upstream bottleneck will slow down the responsiveness and overall performance of the storage. The following series of recommendations were developed from years of architecting and managing high-performance SQL Servers and should be reviewed to determine which practices to apply to your environment.

Target Audience

This best practice guide is intended for system administrators, storage administrators, database administrators, and system architects who design, manage, or deploy database servers and associated infrastructure in datacenters. We recommend that those who implement these best known methods have familiarity with SQL Server database administration and storage systems.

IntelliFlash Storage System Configuration

This section provides a setup checklist and recommendations for optimal installation of IntelliFlash storage systems for performance configuration of enterprise SQL Server workloads.

Controllers and Workload Placement

Each IntelliFlash system contains two active controllers, useful for both maximizing performance and providing high availability with instant failover. Ensure that the disk pools are configured with the maximum number of disks.

If enough additional disks are present, consider separating out multiple pools of disks so that each pool can be assigned to a separate Resource Group, and balanced among the active controllers to improve the performance of the distributed storage.

Make sure that all controller HBAs or NICs are zoned appropriately and connected into the destination server's storage configuration so that all active paths to each controller can be leveraged by the server.

LUNs and SQL Server Objects

Common practice for SQL Server deployments is to separate database objects, such as the data and log files, onto separate LUNs. On more sophisticated deployments, additional separation could be warranted to maintain performance, such as deploying additional LUNs for items such as TempDB data and log files, backups, or other workload files. Remember, storage queueing does not occur just at the storage device itself. Storage

queueing can occur at many layers above the actual SAN and spreading out the workload onto multiple LUNs can help improve the queue constraints and improve performance.

Even virtualized environments can benefit from distributed LUNs for SQL Server object placement amongst individual virtual disks, as additional layers of queueing and multipathing at that layer will also benefit greater distribution for larger SQL Server workloads.

Testing and operational baselines will help you determine the extent of separation necessary to help maintain peak performance for your workloads. The following example shows a significant level of separation for a larger SQL Server workload.

For more detailed recommendations directly from Microsoft, please reference the following URL. https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2005/administrator/cc966412(v=technet.10)

LUN Configuration

For each LUN provisioned for these purposes, make sure that the LUN purpose 'SQL Data' is selected. This purpose preconfigures the block size specifically for SQL Server workloads at a 64KB block boundary, which aligns with Microsoft best practices for SQL Server for allocation unit sizes. The SQL Server workload is more variable than just 64KB block access patterns but aligns well with a 64KB standard. This LUN purpose helps to maintain the best possible alignment with the SQL Server workload and the underlying storage boundaries.

The SQL Server database object contents might be already compressed or encrypted, or could contain types of data that are non-repetitive and less likely to see data savings through either compression or deduplication. Review the SQL Server data footprint, features in use, and LUN properties to determine if enabling or disabling the system data savings features can provide a significant savings, and if not, adjust the LUN configuration accordingly.

For example, the LUNs on this disk pool are leveraging multiple terabytes of SQL Server data, most of which is nonrepetitive in nature. While the data is compressing at a 240% reduction rate, the data deduplication is only at an additional 8% rate.

Most workloads can and do benefit from the data savings, but databases can sometimes be quite different than traditional workloads. Evaluate the type of data and data savings rates to evaluate if the data footprint can benefit from the use of the data savings features.

LUN Snapshots

Storage snapshots are used to create a point-in-time version of the data contained on the LUN that can be leveraged as part of a data protection strategy. They can also be used in a different strategic manner to provide a copy of the data to be re-presented by the storage and re-used by the servers for other tasks.

Certain scenarios in the database administration tasks require very fast response times. For example, if an operating system update was to be found problematic after applying it during a maintenance window, the LUN could be reverted to the state taken at the beginning of the snapshot, and SQL Server would now be back to its earlier version without needing to incur the time and risk of a patch uninstall process. The snapshot rollback process is nearly instantaneous, and the system returns to availability without the added risk.

Be very careful with the placement of SQL Server objects and LUN snapshot rollbacks. If SQL Server has a database's objects widely distributed amongst multiple LUNs, and not all of the LUNs are reverted to an earlier point in time, the database object log sequence numbers could now be out of sync, and that particular database could now encounter problems and could even go offline. All database objects should be part of the same snapshot window so that they can be reverted as a set to maintain database recoverability.

Remember that SQL Server database transactions could be entered into the system during the maintenance window if all application access is not disconnected. If the SQL Server storage snapshot is reverted, these database-level changes could be lost. Items such as new customer purchase details, inventory changes, or security logging data could be lost as a part of reverting a LUN snapshot. Explore the use of reverting just the operating system LUN snapshot while maintaining the state of the other LUNs for the system. The OS could be reverted while the database state remains and protect against the accidental removal of changed data.

Interconnect Multipathing

The use of active multipathing for all protocols of storage connectivity will allow all available storage paths between the database and the storage pool to be leveraged in storage-level communication. This also improves availability by seamlessly rerouting I/O operations to the available active paths if any of the paths were to fail.

By default, only a single interconnect connection between the server and the storage will be used for a given stream of data. SQL Server can request and consume an incredible amount of data quickly, and IntelliFlash systems can deliver upon that level of demand. If the interconnects are the primary bottleneck to high throughput, SQL Server performance can be degraded. If all available interconnects are leveraged for the single I/O stream, the bottleneck to performance shifts away from the interconnects and towards the code being executed within the database engine, thus improving performance in the process.

For Windows and Hyper-V-based servers, the MPIO multipathing feature should be added into the OS. This module will allow for transparent active multipathing for all presented paths to the storage layer, improving the overall performance and reducing the latency to disk.

To install MPIO, add the feature into the Windows platform through the Server Manager. Select Features, Add Features, then select 'Multipath I/O', and confirm the selection.

After connecting your storage to this server, use the Multipath I/O utility under Administrative Tools to select and configure active multipathing for your storage system.

For Linux-based operating systems running SQL Servers, Device Mapper Multipathing (DM-Multipath) should be installed through the distribution's package management system. It will automatically detect multiple paths to the same storage device and manage this connection for you. Edit the /etc/multipath.conf file with your text editor of choice, then start and enable the multipath daemons. Make sure to configure it to launch on system boot.

For VMware-based platforms, ensure that round-robin multipathing path selection policy is configured within the ESXi host configuration for each LUN presented to each host.

Virtualization and SQL Server Storage

Virtualization hypervisors introduce an additional layer between the storage and the SQL Server database. Most hypervisors contain storage-level default values around performance and I/O queueing that are conservatively defaulted to values best suited for historical spindle-based storage and can introduce a significant performance penalty when the hypervisor is under a moderate or greater storage workload. This penalty is clearly visible when monitoring the storage system directly, seeing low latency values, and comparing these metrics against in-guest latency metrics, which show exponentially higher values.

Host Adapter Queue Settings

By default, many Fibre Channel host bus adapters (HBAs) are configured with a low maximum concurrent I/O command queue depth preset to values designed to not overwhelm spindle-based storage. These low queue depths, such as a value of 32, prevent the bulk of the I/O workload from reaching the storage system, so the HBA is artificially slowing down the I/O stream while the system is relatively idle and waiting for more. Review your manufacturer's HBA queue depth settings to determine if it is possible to adjust this queue depth, then validate changes up to the highest possible setting (in reasonable increments) through SQL Server storage benchmarking to confirm latency reduction and I/O performance gains. For IntelliFlash systems, optimal values are between 128 and 256, depending on your HBA and compute platform.

Active Multi-Pathing

The use of active multipathing for storage connectivity does not disappear when a hypervisor hosts SQL Servers. Both VMware and Hyper-V contain default settings that should be adjusted to better accommodate active multipathing of storage commands.

For VMware-based host hypervisors, each VMFS-based datastore contains a path selection policy (PSP). By default, this value is set to Most Recently Used, which selects a single path to the storage. All IntelliFlash datastores should be reconfigured for Round Robin, which should be set by default if the installation procedure is followed.

For Hyper-V-based host hypervisors, the MPIO multipathing feature should be added into the hypervisor and configured appropriately. This module will allow for active multipathing for all presented paths to the storage layer, improving the overall performance and reducing latency.

VM Disk Presentation

While virtual disks might seem different in nature than traditional storage, they are more similar than you might think. In essence, virtual machines (VMs) can connect to storage through a number of different methods.

The most common method is to leverage a virtual disk. A virtual disk is just a file on a shared file system with the hypervisor and usually other virtual machine host servers. Virtual disks are usually placed on a formatted SAN LUN so that more than one host can connect to and mount the same virtual disk(s) as needed allowing the VM to freely move from one host to another.

A virtual disk can be deployed as fully expanded (thick provisioned in VMware or fixed in Hyper-V). A virtual disk can also be deployed as a placeholder (thin-provisioned in VMware or dynamic in Hyper-V) so that the in-guest operating system "thinks" the disk is fully provisioned and available, but the hypervisor has created a tiny virtual disk that expands as data is written into it. A LUN on the underlying SAN can also be provisioned in either manner, and caution must be taken in managing this platform. The combination of the two can cause confusion. A thick-provisioned disk on a data-reducing system can provide different storage consumption numbers depending on which layer an administrator views the metrics from.

For example, a virtual disk is connected to a VM, and from within the guest operating system, thin or thick both appear as fully provisioned, and the operating system does not know the difference. If a virtual disk is thick- provisioned from within the hypervisor, from the hypervisor's point of view, the virtual disk now consumes all of the allocated space on the underlying SAN LUN. From the SAN's point of view, if data savings are enabled on the LUN, the thick-provisioned disk will reduce to a subset of just the space consumed by the data on the virtual disk. The thick- provisioned but unused portion of the virtual disk will reduce to virtually nothing, leaving the data reduction capability of the LUN to perform the normal data reducing techniques to save space on the underlying SAN.

Thin-provisioned disks are viewed by the hypervisor as virtual disks with just the space consumed by data rather than the entire provisioned virtual disk, but the underlying SAN handles the data savings identically. The end result is that the data savings on the LUN will be equivalent.

If the SQL Server availability solution, hypervisor capabilities, or other factors require the use of shared disks directly presented to the SQL Server OS within each VM, SAN LUNs can be directly connected to one or more VMs. One method is to use a Raw Device Map (RDM) if you are on the VMware platform, or a pass-through disk if you are on Hyper-V, to directly present a SAN LUN from the hypervisor to the VM. The VM has no knowledge that it is in a VM, and thinks the storage is directly connected to the VM. This form of disk presentation is the preferred mechanism for shared storage on the VMware platform between multiple VMs if Windows Server Failover Clustering (WSFC) is required and necessary to support the SQL Server high-availability solution, usually SQL Server Failover Clustered Instances (FCIs). On Microsoft's Hyper-V hypervisor, shared VHDX disks will work just fine for shared disks for use with WSFC.

Additionally, if this configuration is more cumbersome than administrators want, or if additional storage-layer functionality is required for operational management, administrators can opt for in-guest iSCSI disk presentation, connected directly through the VM networking to separate and dedicated iSCSI LUNs on the storage system. This architecture requires that the in-guest networking be configured in a redundant and high-performing manner, in addition to the underlying hypervisor, and can cause some additional challenges. However, the use of SAN LUN tooling, such as LUN snapshots, LUN-level replication, or other SAN features can be advantageous to the administrator.

Start with your technical requirements, select the storage presentation that is needed to facilitate the appropriate SQL Server architecture, and keep the architecture as simple as possible to meet or exceed the expectations of the platform.

From a performance standpoint, virtual disks provide a measurable but insignificant amount of performance overhead versus directly connected LUNs to VMs. If at all possible, opt for virtual disks, as these provide the greatest level of flexibility and abstraction away from the physical server architecture.

Additional Disk Controllers

By default, all disks connected to virtual machines start with just one disk controller. The Windows OS loads a driver that allows the underlying operating system access to the connected disks. The driver itself has a concurrent I/O queue depth that limits the number of concurrent I/O operations that can traverse the controller. The hypervisor and operating system both perform I/O coalescing not just by drive, but by disk controller. All major hypervisors allow up to four virtual SCSI controllers to be connected to a single VM.

To improve the maximum available I/O queueing potential from within the VM for I/O-consuming VMs, consider adding additional disk controllers and spreading out the I/O workload amongst multiple disks, each assigned to load balance amongst the available controllers.

Separating out the SQL Server workload to multiple virtual disks allows for portability and flexibility of the various object types, as well as improved I/O concurrency when the potential for bottlenecking exists. The practice of object separation is common within the DBA communities and should be continued with these workloads on the VMware platform.

The virtual disk configuration can then be tailored to this I/O concurrency model of separation of objects. Feel free to modify this configuration according to your standards.

Letter	SCSI	Controller	Purpose
C:	0:0	0	Operating system
D:	0:1	1	SQL Server home
E:	0:3	1	System databases
F:	1:0	2	User database data files (1 of X)
L:	2:0	1	User database log files (1 of X)
T:	3:0	3	TempDB database data and log files
Y:*	0:2	1	Windows page file
Z:**	0:4	3	SQL Server database backup target

Use the Perfmon or other performance measurements to better validate and develop the load-balanced distribution across the available controllers.

* If SAN-to-SAN replication technologies are used at your organization, consider moving the Windows page file to the Y: drive. This virtual disk can be placed on a SAN volume that is replicated much less frequently than the other virtual disks. Performing this modification will allow the constantly changing page file to consume less bandwidth during the replication process. For Zerto replication, you can mark this drive (and potentially the TempDB drive) as a 'Swap drive' and it will replicate the drive structure and folder security, but not the objects, and will (sometimes dramatically) reduce the amount of bandwidth required for replication.

** If local backup targets are normally used for SQL Server database and transaction log backups, consider adding a dedicated virtual disk for the backup target. This virtual disk can be placed on storage geared for capacity and not performance, and the backup traffic can be configured to not utilize the same paths as the path to the production data.

VMware-Specific Paravirtual SCSI Controller

The VMware ESXi™ hypervisor allows for up to four SCSI controllers and four NVMe controllers per virtual machine. The in-guest operating system performs in-guest I/O coalescing not only by drive but by disk controller as well. Increasing the number of available I/O queues in the guest operating system allows administrators to distribute and load balance the workload's storage amongst the available controllers and improve the in-guest I/O storage bottlenecks.

If your SQL Server workload is virtualized on the VMware vSphere platform, you can leverage the performance improvements available in the VMware Paravirtual SCSI (PVSCSI) and NVMe disk controllers. The default disk controller for Windows operating systems is the LSI Logic SAS controller, which has an OS-level queue depth of 32 commands and cannot be changed. The PVSCSI controller has a default queue depth of 64 and can be overridden and adjusted up to 254. More directions for adjusting this in-guest queue depth is found at the following URL:

https://kb.vmware.com/s/article/1017423

Be careful when adjusting this setting, as other storage-level queueing inside the hypervisor architecture could result in additional overhead or even degraded performance if this setting is not adjusted with the other settings in mind. The new NVMe virtual disk controller, available as of vSphere 6.5 and for VMs with a vHardware version at least version 13, has a maximum of 16 queues and a maximum queue depth of 256 with 4K in-flight commands per controller, with up to four of these controllers available for each VM. Each controller can have up to 15 virtual disks attached to it. The goal of the NVMe controller is to leverage the improved queueing of the underlying IntelliFlash system to continue to improve in-guest storage latencies while improving throughput and IOPS with lower CPU associated with I/O operations.

This controller should be tested and leveraged for any SQL Server workload residing on an IntelliFlash system. As with the PVSCSI controller, review your workload patterns and distribute your workload amongst the virtual disk controllers in a load balanced manner.

Operating System

Many facets of the operating system directly impact both SQL Server and the storage subsystem underneath. As such, a number of items are directly suited to performance optimizations.

While SQL Server is normally thought of as a Microsoft Windows-only platform, recently SQL Server introduced some new features which enable it to be deployed on additional platforms. As of the SQL Server 2017 release, the database engine is now fully supported and capable of running on several supported Linux operating system distributions. Supported Linux distributions include Red Hat, SUSE, and Ubuntu Linux, and leveraging modern containers through Docker support, which allows for Apple OS X support.

Server Power Profile

To prevent the Windows Server operating system from reducing the CPU core frequency and potentially reducing the speed of storage processing, set the Windows Server power profile to 'High Performance' instead of the default of 'Balanced'. Balanced mode can reduce the clock frequency during periods of lower CPU consumption, or even "park" CPU cores, both of which will reduce the available CPU speed and subject SQL Server to both lower performance and slower I/O handling within the operating system kernel. 'High Performance' reduces the OS's ability to slow down the platform, which will help maintain performance. Disable the Windows power savings by forcing a 'High Performance' power profile in the Power Options Control Panel utility, or with the following command.

powercfg.exe /SETACTIVE 8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c

Linux distributions also support a higher CPU performance setting. In a terminal console, execute the following command to force the system to the high CPU performance profile.

echo performance | sudo tee /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor

OS-Level Disk Block Sizes

Microsoft recommends the use of 64KB NTFS allocation units to achieve the best SQL Server- layer alignment with the underlying file system.

On the Linux platform, Microsoft only supports either the EXT4 or XFS file systems for use with SQL Server running on Linux. Altering the block size could be more difficult on this platform. EXT4 has been the primary choice of Linux administrators for quite some time. By default, the EXT4 and XFS block sizes are 4KB, similar to NTFS. While the file system format commands support specifying an alternative block size, the primary block sizes are 1KB, 2KB, and 4KB, and none above 4KB. If the Linux kernel itself does not have the precompiled kernel memory paging size defined as a larger value, you will not be able to format a block size larger than the default of 4KB on the x86 platform.

Leverage the built-in IntelliFlash LUN profile for SQL Server workloads to maintain the most optimal alignment between SQL Server and the storage layers. This profile leverages a 64KB block size, which best aligns with SQL Server and other layers leveraging the storage.

Windows Indexing Service

SQL Server rarely makes use of the Windows Indexing service for normal OLTP workloads, except for certain operations, such as the use of FILESTREAM. The indexing service adds additional I/O and CPU overhead while performing the routine indexing operations.

Disable Windows indexing on the SQL Server database object volumes to reduce the possibility of additional storage activity which could add additional IOPS demand on the platform.

Antivirus and Anti-Malware Scanning

Antivirus and anti-malware scanning can increase the demand on storage while also creating a situation where SQL Server might find access to the database object blocked by the OS-level scanner. While their use is important for today's aggressive security footprint, certain database objects can be excluded from the list to eliminate the possibility of object-level blocking that can slow down or cause certain databases to go offline. Follow Microsoft's best practices for all file-level and engine exclusions, available at the following URL:

https://support.microsoft.com/en-us/help/309422/choosing-antivirus-software-for-computers-that-run-sql-server

Enterprise Disk Provisioning

If any enterprise-level storage provisioning and partition management applications are deployed, ensure that the partition offset for logical volumes provisioned from the operating system has been retained. Windows Server 2008 and above began automatically introducing a partition offset for all partitions created by the operating system and the partitions are useful to maintain block alignment with various storage subsystems. Failure to maintain

this partition offset can result in a storage block misalignment, which will cause greater I/O demands and degraded performance. Occasionally, an enterprise-level storage partition management utility can overlook this offset, and silently introduce performance inefficiencies into the storage path. Verify with your storage software, if applicable, that it is not removing this necessary partition offset.

Instant File Initialization

SQL Server, working through the operating system layer, will write zeros to fill the database objects on disk every time a database is created or a database file needs more internal space and performs a growth operation. For some operations, writing gigabytes of zeroes at a time can delay a critical operation, such as a restore. For other operations, such as a data file growth operation, write activity on the database is halted until the operation completes. For these scenarios, a special setting called Instant File Initialization (IFI) can be leveraged to instruct the engine to skip the zero-write operation and complete the operation, which can improve the performance of that operation.

Note: IFI does not apply to database log files.

To grant 'Instant File Initialization' access to the service account that will be used for the SQL Server service, open the Local Security Settings. Expand Local Policies and select User Rights Assignment. Add the domain service account to 'Perform volume maintenance tasks'. Restart the SQL Server service.

Disk Defragmentation Software

Many organizations leverage an OS-layer storage defragmentation solution to 'defrag' the file system. SQL Server, on the other hand, maintains its own fragmentation solutions from within the internal SQL Server operating system, and is managed by routine DBA maintenance practices. The use of OS-level disk defragmentation software on SQL Servers is discouraged, as the vast majority of the space consumed by the SQL Server will not benefit from OS-level disk defragmentation. The defragmentation software will increase the demand on storage without producing any tangible performance improvements by the SQL Server.

Virtual Memory

The virtual memory file (page file on Windows, swap volume on Linux) is leveraged if the operating system requires more memory for application workloads than is physically present. The applications 'think' the memory consumed is actual memory, but the OS has placed these blocks on storage media. The OS then redirects the memory request through storage instead of actual memory, which is many times slower than memory. Active page file usage will cause a significant performance penalty on the application workloads, as well as increased demand from the storage. Ordinarily, SQL Servers rarely leverage the virtual memory file and should not page to storage if the SQL Server memory footprint is carefully managed. Follow your organization's best practices for virtual memory file sizing and placement, but if properly managed, SQL Server should not aggressively leverage the virtual memory file for SQL Server-related activities.

Microsoft SQL Server

Microsoft's flagship database platform, SQL Server, is continuously evolving. At the time of this writing, the most current release version is 2019.

SQL Server has a unique level of interaction with the operating system. SQL Server contains its own small operating system that resides between the SQL Server databases and the underlying operating system. This OS, coincidentally called the "SQL OS," handles items such as CPU threading, process management, storage hardening, memory mapping and management, and relaying the small set of commands to the OS.

Relationship Between SQL Server Storage and Memory

The relationship between SQL Server storage and memory is quite interesting. In the early days of the SQL Server platform, memory was exponentially faster than the average storage underneath the database. To maintain performance of the database engine with commonly used data, the engine used server memory as a storage read cache to keep a copy of this data available for much faster retrieval, and this block of memory is referred to as the buffer pool. Database administrators will set a subset of server memory for the SQL Server buffer pool to leverage for this caching purpose.

Historically, DBAs request a significant amount of server memory to be used for this purpose, which has led to servers, both physical and virtual, with some of the highest memory footprints of any server in the infrastructure.

Certain signs can indicate SQL Server memory pressure. If the SQL Server workload is elevated and the workload is demanding much data for processing, the storage read demand can be elevated. If the SQL Server does not possess enough memory to buffer commonly accessed data, this read demand can increase. The more demand on data, the greater the amount of memory pressure exists in the engine, and the greater the demand is on storage.

However, the SQL Server workload reading more data from disk is not necessarily a negative factor. If the storage performance is sufficient to keep up with the SQL Server demand, having a smaller buffer pool might not be a detriment to exhibited performance, and the workload's performance expectations are maintained.

As the storage performance underneath has improved, the operational practices of leveraging more of the storage performance for data retrieval has not necessary kept up. Typically, DBAs and third-party application vendors will insist on a very high amount of memory for their databases, even if the storage performance is quite high. In many cases, adoption of very fast underlying storage can result in needing a smaller buffer pool memory footprint while maintaining the performance expectations of the SQL Server workload. Reduced memory footprints can reduce the overall up-front cost for these SQL Server platforms, as memory is one of the larger costs for the SQL Server platform.

When virtualization platforms are introduced into the infrastructure, an even more fascinating pattern emerges.

In most environments, the SQL Server CPU workloads are moderate in nature at best, and the VM consolidation ratios are held back by the memory footprint of the individual servers. While vCPUs can be carefully overcommitted in virtualized workloads, the large consumed memory footprint of the VMs cannot and should not be overcommitted. As a result, the VM host server is usually almost at capacity with memory allocations, while the host vCPU scheduling pressure is light enough that additional workloads could be placed onto the host without a performance penalty - if more memory existed on the host. As a result, if the memory on the individual SQL Server VMs can be reduced by introducing fast storage (with appropriate testing for validation, of course), the VM density on the host can be carefully increased while maintaining SQL Server workload performance. For larger scale deployments, this consolidation can reduce the host quantity and exposed SQL Server licensing footprint, potentially saving an organization a significant amount of capital.

Keep in mind that the SQL Server engine shields how it leverages memory for internal operations from the underlying operating system, both Linux and Windows. Windows and VM-level counters are not necessarily accurate in determining if a SQL Server is facing memory pressure. SQL Server DBAs do have a number of metrics to determine the state of memory and any associated pressure from within the database engine, including counters such as buffer cache hit ratio, page life expectancy by NUMA node, memory grants pending, and lazy writes per second. Consult with your DBA to determine the state of memory pressure from within the database engine.

Database Object to Disk Placement

SQL Server object placement can be quite complex for larger deployments. A pyramid of objects soon appears in most SQL Server deployments.

Multiple SQL Server installations, each called an instance, can be installed on an operating system. Each instance has its own unique set of settings and workload properties and can contain one or more databases. Each instance also has its own set of system databases. Each database, at a minimum, contains one file group with one data file (usually suffixed MDF) and one transaction log file (usually suffixed LDF). Each of these files is then configured to reside on one or more logical disks connected to the operating system.

More sophisticated SQL Server deployments can leverage many file groups, each with their own data files, to spread out the objects within the database for logical partitioning, performance advantages, data archiving, or improved recoverability. Each of these files can then be better distributed amongst multiple storage devices or architectures.

Alternatively, some databases are held back due to third-party vendor requirements for maintaining a sub-optimal database architecture because of supportability challenges from the vendor. Make sure that you do not void your software support warranty if you modify a default configuration.

The power of this flexibility is quite impressive. For example, if administrators know that a particular object is not going to work well with LUN-level data savings, such as large BLOBs or columnstore index files, these objects can be placed on LUNs that have data savings features either dialed back or disabled altogether. Archival data that does not change can be placed on LUN(s) that have data savings features configured for maximum data savings, and write caching is thereby reduced. High throughput requirements can be better distributed amongst multiple LUNs, disk pools, and controllers for improved queueing throughout the entire storage stack, and therefore achieve peak performance. TempDB and other transient objects, such as staging tables for batch load and transformation processes, could be placed on LUNs that are not frequently (or not at all) replicated to reduce bandwidth consumption.

Work with the administrators for each layer of the application stack to determine the ideal and most strategic configuration and placement of these objects with the underlying storage aligned to the workload to get the most performance and efficiency out of the entire stack.

TempDB Configuration

The TempDB database, part of every SQL Server instance, is the temporary object storage container for both the system and user databases to leverage. It is used for transient commands such as work tables, data version storage, sorting, and excessive commands. TempDB CPU and storage consumption can be quite high, as some database workloads have very demanding requirements of TempDB. Aligning the TempDB properly with the storage subsystem will improve TempDB performance, which in turn improves the application database performance. To align the TempDB database properly with the storage, the database data files should be spread out so that the SQL Server storage engine will better parallelize the workload, effectively spreading out the SQL Server database I/O access within the engine itself.

For best performance, the modern SQL Server engines will create more than one TempDB database data file as part of the instance installation process. Microsoft recommends that one TempDB data file exist for each CPU core, up to eight files.

If your DBA experiences significant TempDB metadata contention on this server (usually manifesting in the form of SGAM or PFS contention), increase the TempDB data file count up to the number of CPU cores present in the server, starting in groups of four files. As consumption rates increase and/or contention rates grow, distribute these files amongst more logical disks, LUNs, etc. to improve performance.

Instant File Initialization

When on the Windows OS a SQL Server running database file has reached its internal space limits and the database engine needs to expand the database file, the default behavior of the database engine is to allocate an additional block of space defined at the database properties and instruct the operating system to perform the action. The operating system will write zeroes to the new block until it has written over the entire block. (This process is called zero-write.) Then the SQL Server layer allocates this space as part of the database file. For large file expansion operations, this situation can add additional I/O demand and time to the command.

For the SQL Server database file, this process can be changed to improve the performance characteristics and load on storage through a setting called Instant File Initialization (IFI). Enabling IFI will instruct the operating system to bypass the zero-write process and allocate the OS-level blocks directly.

To enable IFI, within the Local Security Policy on the server, expand Local Policies then User Rights Assignment, and enter the service account that the SQL Server service is executing under into the 'Perform Volume Maintenance' option.

Note: SQL Server database transaction log file zero-write operations cannot leverage IFI.

To validate if IFI is enabled successfully, review the SQL Server Error Log for entries just after instance startup.

For more information on the usage of IFI, visit the following URL: https://docs.microsoft.com/en-us/sql/relational-databases/databases/database-instant-file-initialization

SQL Server Database Maintenance

SQL Server databases should perform routine maintenance to improve the efficiency of the data access process, as well as to validate that the data contains no occurrences of corrupt data. These routine (and quite necessary) maintenance tasks include database index and statistics maintenance, transaction log backups, and integrity checks.

All of these operations will subject all infrastructure layers leading to and including the underlying storage to a higher than average load. If bottlenecks in the infrastructure are present, these performance penalties are generally magnified during these operations. It is vital that each layer between the SQL Server database and the storage subsystem be validated for their respective optimal configuration so that the database performance suffers from the least possible penalty while these critical operations execute. SQL Server DBAs should also map the starting times and process runtimes for these processes in an attempt to stagger the runtimes to minimize the impact to the infrastructure.

IntelliFlash Storage Systems and SQL Server Data Savings Features

IntelliFlash systems feature powerful data savings features that provide you with data reduction abilities while maintaining the flexibility to tailor the data savings to the type of workloads placed on the storage system. Many SQL Server features, such as compressed backups, table-level compression, encryption, and column store indexing introduce scenarios where the SQL Server data feature could reduce the data reduction ratios on the storage layer. Many of these features perform data savings based on maintaining performance and not necessarily the most aggressive data savings, such as table-level compression, while others are extremely compressed at the expense of write performance, such as column store indexing.

During some scenarios, the SQL Server layer feature can be quite strategic to leverage, and the storage subsystem can be tuned to maintain optimal performance. In other scenarios such as the use of traditional table-level compression, the historical use of the SQL Server feature can be replaced and both performance and data savings ratios significantly improved through the transparent data savings built into the IntelliFlash systems. We will address some of these features in the following sections.

Native Database Backups

Native SQL Server database backups can leverage built-in compression, which results in a native method of data savings by reducing common SQL Server database backups by over 50% on average.

A sample backup of a 30GB DVDStore benchmark SQL Server database compressed to just 9.8GB, a compression ratio of 3.08 with native SQL Server backup compression.

However, IntelliFlash systems provide greater data savings abilities at the LUN layer than SQL Server's native backup compression, as the SQL Server feature is optimized for backup process speed and not maximum compression ratios.

If the network throughput is limited and is the primary bottleneck to SQL Server-level backup performance and runtime, the use of native database backup compression can mean that less data is written across the network than without leveraging this feature. If this scenario sounds familiar and budgets are constrained, the continued use of SQL Server-level backup compression could be strategic to the business if backup runtime is ranked higher than data savings ratios. Alternatively, when an upgrade to a faster networking infrastructure is recommended, as in modern enterprise platforms, the speed of the network should no longer be a significant bottleneck to overall system performance.

Table-Level Compression vs. LUN Data Savings

SQL Server contains a feature called table compression that allows an administrator to compress a selected database table using either row or page compression. Most tables are at least somewhat compressible. As of SQL Server 2016 Service Pack 1, table-level compression is now available in Standard edition.
The SQL Server compression feature is optimized for speed, but at the expense of a high compression ratio. SQL Server also must use the same CPU cores for the compression and decompression routines as it uses for general database operations. The overhead is small, but certainly measurable. The IntelliFlash systems contain storage processors that are designed to offload this data savings operation in a way that does not impact performance or increase storage latencies. The offload can help on resource consumption rates on the CPUs that are subject to SQL Server licensing by offloading this task to the SAN controllers.

Either database page or row-level compression can be used. Page-level compression is more compressed than row, but at the additional expense of CPU time to perform the compression and decompression routines.

One interesting feature of table compression is that the compressed table data retains its compression when the table data is fetched and loaded into memory, meaning that the engine can store more of this table's data into memory during daily operations. Weigh the benefits of a greater data savings ratio versus more data in the SQL Server buffer pool on the exhibited performance of your applications to determine which is more strategic for your databases.

Every workload's data savings profile is different, so evaluate the actual storage consumption and data savings ratio between a large table leveraging table compression versus an IntelliFlash system LUN configured for data savings and determine which is more strategic for your databases. SQL Server contains a data compression wizard that exists on a table-level that can help estimate the amount of space savings that can be achieved with the function.

Review the data savings ratio on the LUN(s) that the database files reside on to compare the data savings rates. Determine which strategy works best for this particular workload.

Column Store Indexing

Column store indexing is an entirely new method of advanced indexing within SQL Server. While traditional indexes perform their operations starting with each row, and then by a select list of columns, Column store indexes are pivoted to read the column then map the rows. For aggregate numeric functions, such as warehouse-style rollup financial reporting, these indexes provide a significant performance advantage over traditional row store indexing.

Column store indexes are incredibly compressed on the storage media. As a result, data savings ratios for LUNs that these objects reside on can be significantly reduced. Column store indexes are powerful enough where the performance gains are worth leveraging them when necessary, even at the expense of a reduction in the underlying storage's LUN data savings ratios. Explore disabling compression on these LUNs, as the data contained in the index is already significantly compressed. If multiple Column store indexes are used that contain similar or identical data (such as a preproduction copy of a production table from a different database), explore your data savings ratios through deduplication techniques on the LUNs.

Data Protection

Data protection for SQL Server databases is a complex topic, and one not to be ignored. Databases have unique requirements for the synchronization of all database files, the continuity of all items that are present in a SQL Server high availability and/or disaster recovery architecture, and the ability to revert the database to a specific point in time that could be in between a routine backup schedule.

An enterprise solution for database recoverability and business continuity usually consists of both SQL Server and SAN-level features working in combination to leverage the best features of each platform.

LUN Snapshots

While SQL Server can (and should) be configured to take routine transaction log backups, if an error occurs and the database state needs to be reverted, system-based LUN snapshots can be leveraged to streamline this process and reduce the time to complete this operation. Ongoing LUN snapshots can be configured on a schedule to operate in the background, and the LUN can be reverted if a situation arises that requires rapid recovery from a previous point in time. This feature also has less overhead than VM-level snapshots.

Useful scenarios for this feature include rollback for SQL Server patching and application upgrade situations to protect against if the operation fails and the system and/or database state must be reverted to a previous state.

Keep in mind if a LUN snapshot is reverted, the database contents revert as well, so ensure that no data has changed or that any records that have changed are safe to roll back, or back up the database before the system state is reverted. If multiple LUNs are used for a single SQL Server system, ensure that all LUNs are snapshotted and reverted to the same spot, as restoring to different points in time can cause corrupt databases or inconsistent system states.

LUN Snapshots for Pre-Production Database Refreshes

While the rollback process is the most widely known application of SAN LUN snapshots, additional purposes can be quite advantageous to both systems and database administrators. Common DBA tasks can be streamlined. For example, a very large database is routinely backed up from a production system and restored into a pre-production SQL Server for use in development tasks. An additional copy of the data is restored onto a reporting server nightly for use by the business to offload this workload from the production database. The time taken for the backup and restore process can be quite significant, and the tasks impose a performance drain on the systems.

SAN LUN snapshots can be orchestrated to streamline this process. The LUN containing the primary production database, regardless of whether the workload is virtualized or still physical, can be snapshotted. The snapshot can then be presented as a new LUN to the infrastructure. The reporting and preproduction servers can then connect to the LUN, mount the storage, and attach the databases. The refresh process can be orchestrated by taking the databases offline momentarily, refreshing the LUN snapshot, then bringing the databases back online. The entire process to refresh both databases can be performed in seconds, whereas the traditional backup and restore process could take so long that it cannot finish in normal overnight hours.
Many layers of the infrastructure can also create snapshots, such as virtual machines and SQL Server databases. Each allow for a point-in-time snapshot of that object from the perspective of that particular layer. Not all snapshots are created equal, and many have substantial limitations to their use and effectiveness. Snapshots should be part of a multi-silo data integrity strategy and evaluate the snapshot capabilities of the individual layers to determine the best solution to a given challenge.

Volume Snapshot Service

The usage of Windows Volume Snapshot Service, or VSS, for SQL Server-level backup integrity is quite interesting, and lies contrary to most other applications on the market. SQL Server, like all other ACID-compliant relational database engines, requires that any data being written to disk have that write operation hardened to disk before the transaction is considered complete. The database engine locks and has exclusive access to the database data and log files. The only way that other programs can access these files is through the SQL Writer Service, which allows other programs to copy these files while SQL Server is running. If this service is not active, no other programs can access these files.

VSS is leveraged by certain backup agents to temporarily halt I/O operations and flush any write operation that is pending to storage so that the backup is consistent and helps to ensure that all files are recoverable on the backup set.

The VSS file system quiesce command that 'freezes' and 'thaws' I/O operations ensures that if a backup routine iterates through the logical disks, the storage operations are briefly suspended to allow the backup process to get a consistent copy of the data on storage. If this process is currently in use on your SQL Servers, you might see entries in the SQL Server Error Log that contain similar entries to the following.

From a SQL Server perspective, any data that not committed to storage is not considered a committed and hardened transaction. It bypasses the Windows-level I/O caching features of the Windows OS and writes directly to storage media. As a result, all database-level transactions are consistent without the need for a VSS file system command.

However, to get a copy of the database files for a server-level backup, the SQL Writer service does allow for the backup process to work with SQL Server to get a copy of the files to fulfill the backup process.

For most environments, leverage SQL Server for database-level and transaction log-level backups, and a different utility to backup the server (physical or virtual). Recovery becomes a two-part process: restore the physical/virtual server, and then restore the databases. Disable SQL Server-specific functionality in the backup utility so that it does not interrupt the SQL Server log sequence as part of the backup process.

If your third-party backup utility can perform a point-in-time restoration of SQL Server databases and apply transaction logs, validate the use of this functionality and potentially leverage this utility for all server and database- level backup and restoration processes. Only go this route if the software provides all of the functionality required by the organization for backups and restores, or else the business continuity strategy could be at risk.

Benchmarking and Monitoring

Any change made to any part of the SQL Server infrastructure stack should be validated with a benchmark of the system state after the change and compared to an ongoing operational baseline of that component to validate that the system change resulted in a net positive outcome. Many tools claim to provide in-depth visibility into a number of layers of the IT stack, but most provide limited depth for the areas outside of their core focus.

At the time of this writing, there exists no reliable synthetic database load testing tool that can produce a storage test and represent your database workload. For storage-level testing, Microsoft DiskSpd provides a solid mechanism to simulate certain individual usage patterns on disk to help establish a baseline. However, strategic use of SQL Server Distributed Replay can help you record a production workload and replay it against a test system to best simulate that workload's I/O footprint and establish the baseline of that workload.

DiskSpd

Microsoft DiskSpd is the replacement to SQLIO, Microsoft's former storage load testing utility. It is an open-source command line utility that allows administrators to simulate certain disk workload patterns on disk, including read or write, random or sequential (or percentages thereof), test intensity per thread, number of worker threads, Windows write caching enabled or disabled, and other such parameters. An example of the command line interface parameters and output workload is as follows.

An output histogram is then generated for the storage response times by percentile for both the read and write portions of the testing.

Keep in mind that even though you can simulate certain individual workload types with DiskSpd, the workload file itself that DiskSpd uses is not remotely equivalent to a SQL Server data and/or log file. A SQL Server data file can contain any number of patterns, including repetition in small portions of data, and areas of free space to allow data to be inserted into areas of a database other than the end (to reduce fragmentation). A SQL Server log file consists of a tightly-packed stream of individual transactions recording changes in the data. The DiskSpd workload file is either a repetitive string of the alphanumeric characters available by default, or a completely random string of gibberish if the -Z entropy command is leveraged by the administrator and is not representative of a true SQL Server data or log file.

This utility can provide an idea of the performance characteristics of one type of a storage-bound workload and should be part of the evaluation process of any new storage device to provide a baseline for that type of test that can be used at a later date.

For more information about DiskSpd, please visit https://aka.ms/diskspd.

SQL Server Distributed Replay

While synthetic tools are useful for delivering a very specific type of workload to disk, no load test software will simulate your application's database workload better than the workload itself. SQL Server Distributed Replay is a feature first introduced in the SQL Server 2012 release that allows for an organization to record a production server's SQL Server commands, prepare a separate test system, restore a copy of the database(s), and replay the commands captures from the original server to simulate that specific workload.

Distributed Replay is incredibly useful when looking to simulate an actual workload on new servers, validate the impact a system change will make on resource consumption, or develop an operational baseline for a system. It is valuable for establishing a storage performance profile so that metrics such as latency, IOPS, peak throughput, and LUN settings and the resulting data savings ratios can be evaluated and modified.

For more information on SQL Server Distributed Replay, visit the following URL: https://docs.microsoft.com/en-us/sql/tools/distributed-replay/sql-server-distributed-replay

DevOps

The topic of DevOps is quite new to DBAs. DevOps is a newer philosophy on integrating the various development personnel to streamline coordination between the staff, automate tasks and software releases, and improve the quality and timeliness of software development releases. Improving the speed of the underlying storage can contribute to successful DevOps strategies in a number of ways, including:

- Rapid production to preproduction clones and refreshes of application databases for development and testing purposes, which can reduce the necessary time to just seconds
- SAN LUN snapshot rollback time for reverting failed code deployments
- Rapid storage deployment of new VMs and containers through system awareness with the hypervisors (VMware VAAI-specifically)
- Significant API for automating numerous system-based tasks, such as LUN snapshot management, snapshot refreshes, LUN creation, and LUN deletion

Additional References List

DiskSpd https://gallery.technet.microsoft.com/DiskSpd-A-Robust-Storage-6ef84e62

Microsoft MPIO for Windows Server iSCSI https://www.petri.com/using-mpio-windows-server-iscsi-initiator

Ola Hallengren SQL Server Maintenance Solution https://ola.hallengren.com/

SQL Server Antivirus Exclusion Recommendations https://support.microsoft.com/en-us/help/309422/choosing-antivirus-software-for-computers-that-run-sql-server

SQL Server Columnstore Indexes https://docs.microsoft.com/en-us/sql/relational-databases/indexes/columnstore-indexes-overview

SQL Server Distributed Replay https://docs.microsoft.com/en-us/sql/tools/distributed-replay/sql-server-distributed-replay

SQL Server on VMware Best Practices Guide https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/solutions/sql-server-on-vmware-best- practices-guide.pdf

SQL Writer Service https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/sql-writer-service

VMware PVSCSI Controller Queue Depth https://kb.vmware.com/s/article/1017423

Disclaimers

IF-BKM002–1219–01