WESTERN DIGITAL PRESENTS

THE GORILLA GUIDE TO...®

Modern Storage Strategies for SQL Server

David Klee

INSIDE THE GUIDE:

- Optimize your SQL Server database
- Consolidate OLTP and OLAP applications with NVMe
- Virtualize SQL Server databases with NVMe

HELPING YOU NAVIGATE THE TECHNOLOGY JUNGLE!

ActualTech Media
www.actualtechmedia.com

In Partnership With

Western Digital®

THE GORILLA GUIDE TO...

Modern Storage Strategies for SQL Server

AUTHOR
David Klee

TECHNICAL REVIEWER
Ed Leighton-Dick

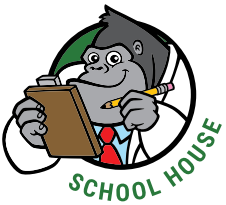
EDITOR
Keith Ward, ActualTech Media

LAYOUT AND DESIGN
Olivia Thomson, ActualTech Media

ACTUALTECH MEDIA
Okatie Village Ste 103-157
Bluffton, SC 29909
www.actualtechmedia.com

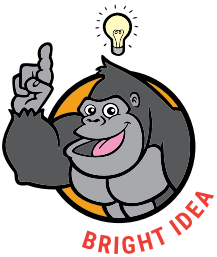
ENTERING THE JUNGLE

CALLOUTS USED IN THIS BOOK



The Gorilla is the professorial sort that enjoys helping people learn. In the School House callout, you'll gain insight into topics that may be outside the main subject but are still important.

This is a special place where you can learn a bit more about ancillary topics presented in the book.



When we have a great thought, we express them through a series of grunts in the Bright Idea section.

Takes you into the deep, dark depths of a particular topic.

Discusses items of strategic interest to business leaders.

ICONS USED IN THIS BOOK

DEFINITION

Defines a word, phrase, or concept.

KNOWLEDGE CHECK

Tests your knowledge of what you've read.

PAY ATTENTION

We want to make sure you see this!

GPS

We'll help you navigate your knowledge to the right place.

WATCH OUT!

Make sure you read this so you don't make a critical error!

CHAPTER 1

Introduction

Data is starting to be treated as "the new oil" by organizations worldwide and is largely treated as the most important part of an IT infrastructure.

Data is not exactly a scarce commodity like oil, but it is a gold mine for organizations that can strategically make the best use of the data within their company.

Data is the fuel for improving the business through modern data science, business intelligence, decision support systems, customer self-service, and organizational efficiency.

Data is flooding into businesses from all sides. IoT devices pump data into the business at ever-increasing rates. End users are going mobile and accessing systems more frequently, from more locations, and with increased expectations on availability and performance.

As such, the volume of data retained and accessed by today's organizations is exploding at an unprecedented rate. Business expects the IT department to keep data online and accessible indefinitely. Organizations are finding more methods to query and mine the data, and demands on the data platform have never been greater.

All of this data has to be stored somewhere, somehow, and companies are increasingly struggling to keep up. To start with, there must be space on which to store the data.

There's also the matter of bandwidth. The platform must be fast enough to return the data to the users at the speed of business. In the modern era, that means immediate access; no delays acceptable.

In addition to being fast, the data must be resilient, and readily available in spite of hardware failure, natural disasters, or human error.

It's a lot to ask of data and its support systems, which is why these platforms have historically been very complex. Full-time infrastructure and database administrators spend careers protecting the data and the infrastructure around it.

Each system component requires planning and testing to make sure that a failure won't hurt the business. Plus, each layer requires in-depth architecture and careful planning to ensure that the stack will perform to its fullest potential. This means best-of-breed components, cutting-edge techniques, and highly trained staff are needed to engineer these platforms.

Full-time infrastructure and database administrators spend careers protecting the data and the infrastructure around it.

As a result, the continuous investment by the business is normally very high. After all, the data protected and the systems that allow access to the data make up the core of most businesses. If the data is unavailable or goes missing, the business could be in big trouble. In worst-case scenarios, businesses have failed because of the loss of data.

This is why it's the administrator's responsibility, first and foremost, to ensure that data is not lost, no matter what event occurs. Ensuring that the data is presented to the application in a timely manner is usually the next priority.

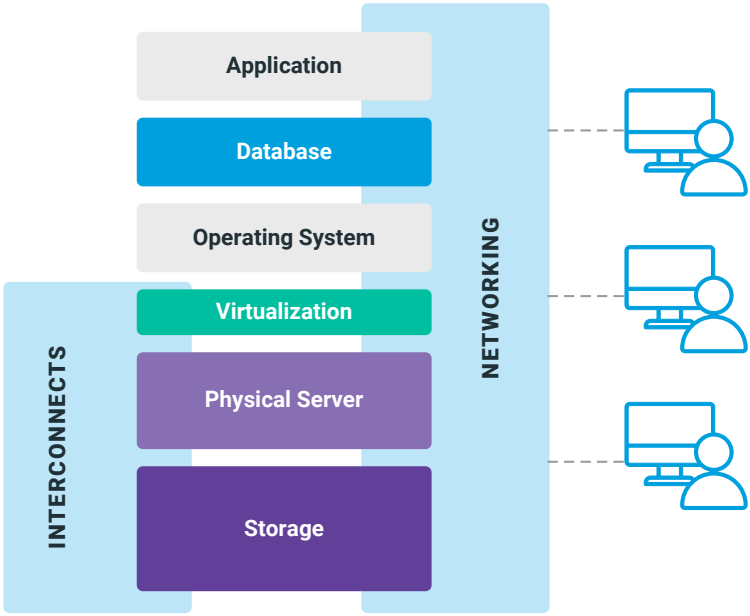


Figure 1-1: The layers of a data center.

Data and Infrastructure Intersect

As shown in **Figure 1-1**, many layers exist in the core of these data platforms. One traditional challenge with these complex infrastructures is how the layers interact with each other, and how they are managed as a whole. A best practice for one layer might be a horrible idea for another layer. An application-specific practice for a layer might hurt the effectiveness of others.

Learning how these layers interact, complement each other, and potentially compete with each other is vital to architecting the modern on-premises data center.

Adding further complication into the mix is the continued growth of the public cloud. Permutations of public cloud offerings can complement and even replace some of the layers in this infrastructure.

Applications and/or database-as-a-service (DBaaS) can remove the requirement to manage the operating system underneath them. Applications, databases, and virtual machines (VMs) can be replicated to the public cloud for disaster recovery (DR) purposes, or even replace the need to have these components on-premises altogether.

The Storage Layer

The bottom layer in this datacenter model (**Figure 1-1**) is the storage subsystem. This device is arguably the most critical layer in the data center. It physically stores the data onto persistent storage, which is usually made up of spindle-based magnetic, solid-state flash, or NVMe™ flash based hard drives.

Usually, SANs are in place for simplifying management and reducing the footprint in the data center. SANs are designed to be robust and protect the data at all costs, and to protect against failures. Modern SANs have added extreme performance characteristics on top of the resiliency of the core platform.

While the storage can be contained within a physical server, this configuration presents a point of failure in that physical server. Instead, it is often deployed as shared storage in a Storage Area Network, or SAN.

Usually, SANs are in place for simplifying management and reducing the footprint in the data center. SANs are designed to be robust and protect the data at all costs, and to protect against failures. Modern SANs have added extreme performance characteristics on top of the resiliency of the core platform.

Historically, the SAN was the slowest layer of the infrastructure stack. SANs usually measured response time in milliseconds, while server

memory or CPUs are measured in microseconds or even in nanoseconds. Magnetic spindle-based storage medium was used to store the data.

More recently, solid-state drives (SSDs) were introduced, which provide order of magnitude improvements in the peak IOPS (Input/Output Operations Per Second) and reduced latency of I/O operations over spindle-based storage.

Most recently, nonvolatile memory express (NVMe) was introduced to replace the protocol and connectivity bottlenecks that emerged once the spindle was replaced with flash memory cells. With NVMe, storage latency continues to drop, peak IOPS climbs, and the concurrent I/O requests are greatly improved over SSDs.

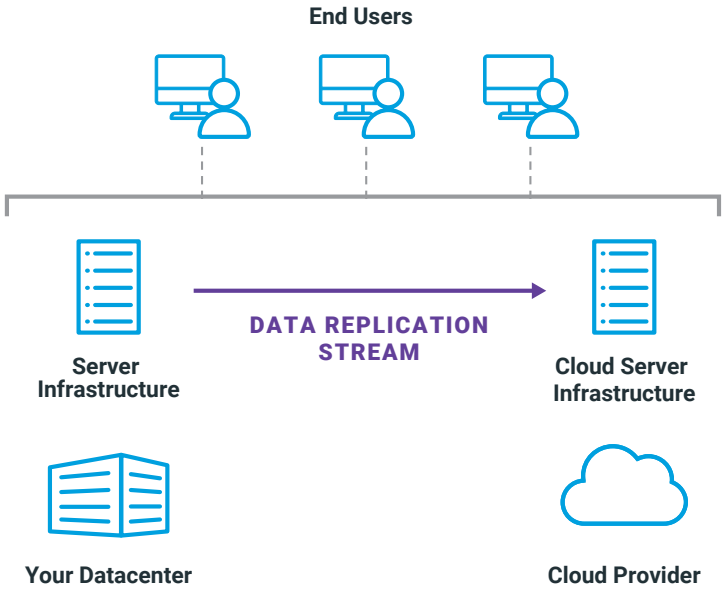


Figure 1-2: A hybrid cloud data center.

Today, modern SANs perform storage operations in microseconds, with capacities well beyond the terabyte range. While still not as fast as server memory, enterprise SANs with NVMe storage are closing the gap on performance.

The Physical Server

The next layer is the physical server, which contains the rest of the compute hardware (CPU, memory, and interconnect adapters). This layer is where the actual work of the application is performed. The data is still stored on the SAN.

Virtualization

The physical server normally contains some type of virtualization, which is an added layer treated as an extension of the physical server.

While not mandatory, most data centers use virtualization to encapsulate workloads and improve the agility of the data center to increase responsiveness to an ever-changing world.

This layer, called a *hypervisor*, allows multiple compartmentalized operating systems and their respective applications to coexist independently on the same hardware. The VMs submit requests for compute resources, and the hypervisor coordinates the access to the physical compute resource to fulfill the request.

By allowing more than one of these VMs to run on the same physical server at the same time, the efficiency of the environment improves, and things such as the actual physical server count, rack space, power, cooling, and interconnect cabling all get reduced. This equipment reduction saves overall data center costs.

While not mandatory, most data centers use virtualization to encapsulate workloads and improve the agility of the data center to increase responsiveness to an ever-changing world.

Interconnects and Networking

The physical server must be connected to the SAN and to other servers through an interconnect, usually fiber optics or high-speed networking, which provides one or more layers of interconnect switching.

Data must be stored in a system designed to store and retrieve the data efficiently. The database server is the gateway to the critical business data stored on the SAN. Microsoft's flagship data platform, SQL Server, is one of the most widely used database engines in the world.

Data centers commonly have separate, dedicated equipment to handle storage communication and application communication so that one doesn't hinder the performance of the other.

The Operating System and Applications

The logical server, be it physical or virtual, contains an operating system where different application servers are installed onto. Some of these servers contain applications which present the data to the end users, such as a web server, while other applications servers are the database servers that contain the data, such as a SQL Server.

Database Servers

Data must be stored in a system designed to store and retrieve the data efficiently. The database server is the gateway to the critical business

data stored on the SAN. Microsoft's flagship data platform, SQL Server, is one of the most widely used database engines in the world, and is the focus of this guide.

Modern Data Center Challenges

Data center topologies are amazingly complex. No two data centers are the same, but all face common challenges with storing and retrieving large volumes of data. These challenges include:

- **Data age.** Businesses demand that a lifetime of data remains accessible. This means the data must be online and accessible at all times.
- **Cost.** Data volumes are growing exponentially; but contrary to popular myth, storage prices aren't falling. SANs are sometimes prohibitively expensive, and are almost always the most expensive component of a data center.
- **Complexity.** The complexity of managing data increases dramatically as the volume and performance demands increase. When this happens, the agility and flexibility in the environment is reduced, and management costs skyrocket at an unsustainable rate.
- **Scalability.** Infrastructure bottlenecks can exist in any data center. As the volume of data goes up, certain components just can't handle the load, and pain points start to appear. Maybe the applications slow down, or the storage fills up, or the infrastructure becomes unreliable. The business suffers from the lack of predictability in the environment.
- **Data sprawl.** Multiple copies of data can litter the environment. For example, development and test copies of data increases the volume of data an organization is forced to maintain, and it drives up the cost of data storage even more.

- **Interaction.** Processes and management techniques for one layer can conflict with the best practices for another layer. The impact of these competing processes can be performance penalties, more complex configurations, or even worse, system outages.

Complementary Technologies

A working knowledge of the basics of each layer is critical to properly understand how to leverage each layer and improve the reliability and performance of the others.

Two of the most challenging layers—storage and the database—lie at the heart of the application stack. Fortunately, these layers don't have to be at odds with each other. Properly architected, these two layers can be quite complementary and present a technical solution that improves the business's chances at succeeding in today's competitive world. Each layer and how they interact will be discussed in depth in subsequent chapters.

Up Next

With an understanding of the ways the data and storage layers interact in the modern data center, let's move on to discuss some of the intricacies of the database engine and the ways it uses the storage layer to function.

CHAPTER 2

SQL Server I/O

At the heart of SQL Server are two layers: the query and storage engines. The query engine interprets an application's request for the retrieval of specific data. The storage engine then takes this specific request from the query engine and fetches the appropriate data from disk.

The storage engine uses a portion of Windows Server® operating system memory as an I/O read cache to store the working set of data during processing. It also uses this memory as a buffer to improve the performance of queries that fetch frequently-read data.

A normal SQL Server can be one of the largest I/O hogs in your server environment. A SQL Server with a poorly designed database, inefficiently constructed queries, or improperly managed maintenance strategies can overwhelm your storage; this can cause the response time of the I/O operation, or latency, to skyrocket, slowing down the operation.

SQL Server I/O Performance

A normal SQL Server can be one of the largest I/O hogs in your server environment. A SQL Server with a poorly designed database, inefficiently constructed queries, or improperly managed maintenance strategies can overwhelm your storage; this can cause the response

time of the I/O operation, or latency, to skyrocket, slowing down the operation.

Why?

Because when a database receives a request for a certain set of data, the query engine determines what data to fetch and instructs the storage engine to retrieve it. The storage engine then goes and carries out the request.

If the query asks for more data than it needs, such as unnecessary table columns, the storage engine has to retrieve more blocks of data from the storage.

Sometimes the request could require certain filtered data, such as the data between two dates, or the data that pertains to a specific purchase order number or customer. That means the query engine might have to scan all the data to find the specific records if the indexing strategy is sub-optimal.

The response time of the SAN is most critical to the overall runtime of the query, and the lowest response times from the SAN result in a substantial runtime improvement of that operation.

Sometimes the query could request the same block of data from disk multiple times. Other times, different tables might be joined to fulfill the request, such as a request to get all orders that a particular customer has placed in the previous year.

Any of these operations can cause immediately elevated I/O demand. Whenever these operations are executed, a single query can cause gigabytes (or more) of data to be read from disk but end up only needing to return one or two records. The response time of the SAN is most critical

to the overall runtime of the query, and the lowest response times from the SAN result in a substantial runtime improvement of that operation.

SQL Server Workload Patterns

Each of these requests from the SQL Server storage engine can generate a different workload pattern on the storage layer. These patterns impact your storage performance if the infrastructure configuration is sub-optimal for the workload type. Fortunately, similarities exist between workload types, and patterns will emerge from the workload. The storage array configuration can then be tailored to best fit the workload properties.

Figure 2-1 demonstrates some of the workload patterns that SQL Server will use as it accesses the storage layer throughout normal daily operations.

WORKLOAD TYPE	BLOCK SIZE	DISK PATTERN
OLTP	8KB / 64KB	Random read/write
OLTP Read Ahead	8KB – 1MB	Sequential read
OLTP Table Scans	512KB	Sequential read
Transaction log commit	60KB	Sequential write
Transaction log read	120KB	Sequential read
Database backup	1MB	Sequential read
Bulk load	256KB	Sequential write
SSAS Cube Workload	32KB	Random read/write

Figure 2-1: Workload characteristics.

Keep in mind that the intensity of these workloads is largely based on the application demands of the database, and can be quite severe at times. Some of these demands include:

- **Routine maintenance operations.** SQL Server databases need periodic index and statistics maintenance to stay optimal. This index maintenance can create an intense window of random read and write activity.
- **Routine database integrity checks.** These create a large amount of read activity.
- **Other tasks.** Activities such as nightly data loads and analytics processing can generate significant workloads.

These tasks, which come from seemingly every direction, can bury even the highest-performing storage if not properly managed.

TASK	START TIME	AVG. END TIME	FREQUENCY
VM-level backup	2 AM	2:30 AM	Nightly
Database full backups	3 AM	4:10 AM	Nightly
Database transaction log backups	12 AM	(20 sec)	Every five minutes
Database index maintenance	10 PM	11:30 PM	Nightly
Database integrity checks	6 AM	9:20 AM	Every Sunday
Data bulk load	6:45 PM	7:30 PM	Nightly
Inventory refresh	7:45 PM	8:20 PM	Nightly

Figure 2-2: Routine scheduled tasks.

You should work with your database administrators (DBAs) to map out each one of these routine activity windows—per server—and map it across the entire environment. You can see an example of these activities in **Figure 2-2**.

The activity patterns on the SAN can be directly traced back to these windows of activity, and the results might surprise you. DBAs might not realize that workload runtimes could be overlapping, or that the concurrent tasks might compete for I/O and slow down the overall performance of the environment.

Architecting the SQL Server I/O Stack

Let's dive deeper into the infrastructure between the SQL Server query and the depths that it must go to get the bytes of data from disk.

The access times for a block of data, called seek times, of spindle-based storage are measured in milliseconds, usually in the 4-11ms range. A spindle-based disk can perform between 100-250 IOPS, based on the rotational speed and the type of disk activity being performed, such as read/write or random/sequential access.

As shown in **Figure 2-3**, the bottom layer is the storage layer. This device consists of spindle-based magnetic disks, solid-state flash memory disks, or a combination of the two. These disks are logically grouped into one or more pools of the same type of disk, configured in a fault-tolerant manner so they can withstand drive failure without losing data.

The performance characteristics of the storage pools vary greatly depending on the type of storage device in the pool.

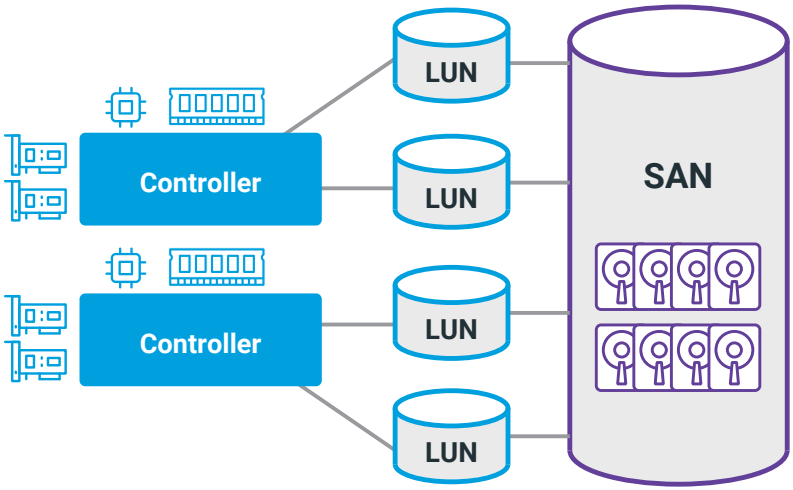


Figure 2-3: A common storage array architecture

Spindle-based storage uses one or more spinning magnetic disks, called a platter, to store binary information on. The data is arranged in a circular row, called a track. The tracks are then grouped into sectors.

An arm with an electromagnetic device on the tip can read or write data by aligning itself with the correct track and waiting for the disk to spin to bring the block of data to it so it can be accessed.

The access times for a block of data, called seek times, of spindle-based storage are measured in milliseconds, usually in the 4-11ms range. A spindle-based disk can perform between 100-250 IOPS, based on the rotational speed and the type of disk activity being performed, such as read/write or random/sequential access.

A solid-state disk (SSD) offers the same functionality to store data as a spindle-based device, but rather than a spindle to store the data on, it leverages interconnected nonvolatile flash memory semiconductor chips.

SSDs have a much lower latency to access a block of data. No mechanical delay exists to access this block, and the drive can access the data at nearly the speed of the electrons in the device.

SSDs usually have the capacity for 32 - 64 concurrent commands. As such, the concurrent IOPS rating is usually in the 70,000 - 100,000 IOPS for a 4KB block size from disk, all while maintaining a latency of well below a single millisecond. That's a major leap in performance!

The next evolution in storage came with the adoption of non-volatile memory express (NVMe) disks. NVMe disks are similar to flash in that there are no moving parts, but they have a different controller interface and storage protocol designed to improve performance over a much higher interconnect bus.

Common interconnect architectures can leverage fiber optics or traditional networking layers. Furthermore, a controller can have more than one interconnect port on the device. The additional ports can help to load balance across the available end-to-end connection paths, and can also reduce downtime if a port fails.

NVMe-based storage offer improved latency over SSDs, and the storage latencies are measured in microseconds. They also offer much higher peak throughput capabilities, which in turn raises the maximum IOPS. They also support a maximum of 64,000 commands in a single message queue, with a maximum of 65,535 I/O command queues. The random access time of these disks make them the fastest possible storage devices on the market today.

From there, one or more SAN controllers act as an intermediary between the outside world and the storage disks. Their job is to accept an I/O request, send the request to fetch the block of data from the disk pool, and return this data out the way it came.

These controllers contain network or fiber optic adapters that connect the controller to the outside world, and contain their own CPUs and cache memory, which act as an I/O read-and-write buffer to boost the overall performance of the array.

The SAN controllers connect into the storage interconnect switches. This layer allows the storage controllers to be connected to many servers or other devices on the network.

The addition of virtualization hypervisors, such as VMware vSphere and Microsoft Hyper-V, into the data center was a seismic shift in data center technology.

Common interconnect architectures can leverage fiber optics or traditional networking layers. Furthermore, a controller can have more than one interconnect port on the device. The additional ports can help to load balance across the available end-to-end connection paths, and can also reduce downtime if a port fails.

These ports are also rated for a maximum data transmission speed. These speeds are important, because if the SAN is faster than the interconnect's end-to-end overall path, the performance of the storage—as the endpoint experiences it—might not reach its full potential and could cause slower-than-expected performance.

At the other end of the interconnects are the physical compute servers. The interconnects connect into these physical servers through devices such as host bus adapters (HBAs) in the case of fiber, or network interface cards (NICs) for Ethernet connectivity. As with the SAN, more than one can be used to provide greater performance and resiliency against equipment failure.

These compute nodes also contain the processing portion of the infrastructure stack: CPUs and memory. The CPUs are configured with

multiple CPU-processing cores on a single chip, and multiple sockets for these chips can be used in tandem in the server. Memory is then placed next to each of the CPU sockets to improve the performance. This configuration creates non-uniform memory access (NUMA) nodes, or a grouping of CPU sockets and a bank of memory.

The Virtualization Layer

The addition of virtualization hypervisors, such as VMware vSphere and Microsoft Hyper-V, into the data center was a seismic shift in data center technology. Servers became highly consolidated, SAN technologies became more ubiquitous and denser, and the workloads continued to grow.

It wasn't without its drawbacks, though. As the workload demands grew with the business, the storage density and performance increased, but many storage settings inside the hypervisor had been left at default values from years prior. These default values can cause significant headaches for administrators.

Consider different storage latency monitoring tools, for instance (**Figure 2-4**). Take a virtualized SQL Server currently residing on a NVMe-based SAN. If a SQL Server monitoring tool reports that "long I/O alerts" are occurring, DBAs are rightfully nervous. A long I/O alert within the SQL Server database is when the database engine triggers an alarm if it experiences an I/O operation that lasts longer than a full 15 seconds.

The primary challenge of virtualized SQL Server comes from how virtualization creates a shared-everything platform of compute resources. Applications like SQL Server don't like to share. They're extremely sensitive to any delay of the infrastructure underneath.

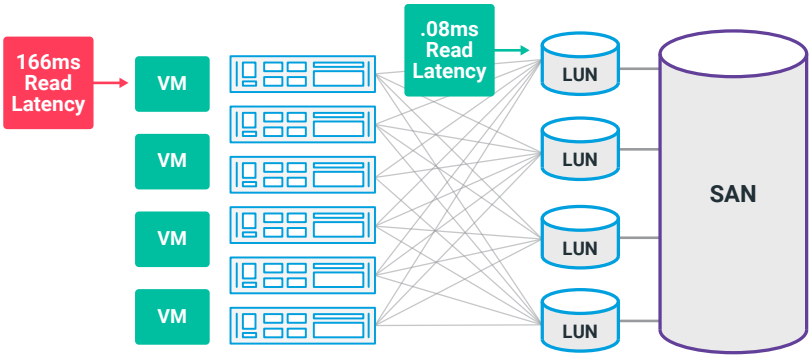


Figure 2-4: Storage read latency and monitoring tool perspectives.

The storage administrator then reviews the performance metrics on read storage from the LUN where the SQL Server resides, using the tooling provided by the SAN vendor. From this perspective, the latency metrics are showing extremely low latencies, and reports back that everything looks great.

This is important because multiple I/O queues exist inside the hypervisor. If one of the queues is set to a default value that's much lower than the demand coming from the source server, this queue depth difference can trigger a significant performance penalty from that point onward.

Because of this, items such as HBA queue depths, shared datastore queues, and disk controller queue depths must be analyzed to determine the default values and the optimal configuration for the underlying storage.

SQL Server and Virtualization

Virtualization is everywhere in data centers worldwide, so chances are that some or all of your SQL Server are now virtualized. That's a good thing. The added benefits to the data center are tremendous, and the agility that virtualization brings to your enterprise is game-changing.

However, virtualizing SQL Server requires quite a bit of tender loving care to ensure that database performance stays strong.

The primary challenge of virtualized SQL Server comes from how virtualization creates a shared-everything platform of compute resources. Applications like SQL Server don't like to share. They're extremely sensitive to any delay of the infrastructure underneath.

Plus, infrastructure admins usually don't have much hands-on experience with SQL Server. Practices that work great for data center consolidation can greatly penalize the business-critical application performance. This presents a thorny problem.

Care must be taken to minimize the impact of multiple VMs trying to access the same physical compute resources through the hypervisor. If

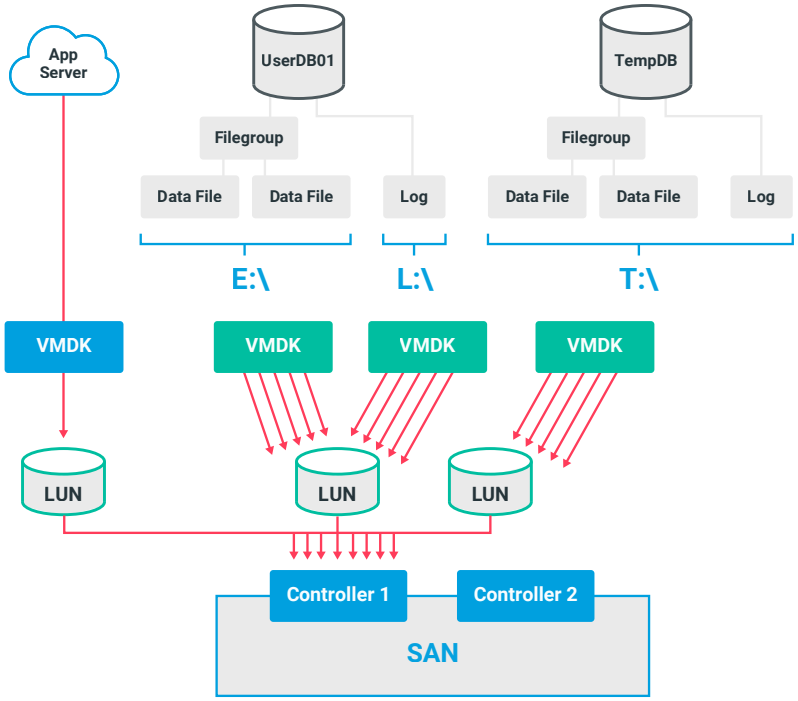


Figure 2-5: Heavy I/O traffic causes contention.

one VM consumes all resources on a host, the impact on the other VMs on the same host can be quite negative.

For example, if one SQL Server is running a database integrity check, which is very I/O intensive, the SAN LUN and controller that the data is traversing could become quite active and busy.

The added load could cause the physical HBA or network adapter to reach its maximum throughput. In some cases, no room exists on this interconnect for the other VMs to perform their normal duties. As a result, the other VMs' I/O requests are backed up in the numerous queues inside the hypervisor, and the application experiences the impact of this in the form of high disk latency. See **Figure 2-5**.

SQL Server Instance

The SQL Server installation, or instance, is a logical hierarchical structure, as shown in **Figure 2-6**.

A SQL Server instance can contain one or more databases. Each database contains one or more logical file groups, which themselves contain one or more data files which store the information in tables. These files can be placed on one or more logical drives or mount points inside Windows Server.

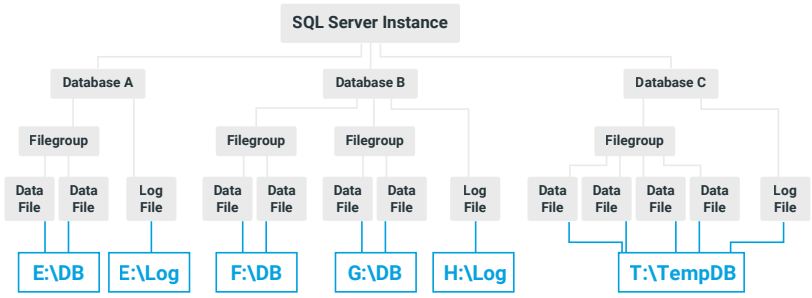


Figure 2-6: SQL Server database structure.

Slow round-trip latencies in the storage layer's response to SQL Server's I/O requests can dramatically slow down the performance of the database queries.

In addition to the database data files, a transaction log file is used to store the record of times that are added into the database, or store the details of data that change or are removed.

Any change into the database is written to this log file, and the database command will not finish until this transaction has been written to disk. The database depends on this log to provide a record of what has changed and when, and it must be in perfect synchronization with the data files for the database to operate normally.

If the storage layer cannot write this stream of data in the log file fast enough, the performance of database change operations is sure to suffer.

A database specially designed for use by the instance for temporary objects, called *tempdb*, is also present on each instance. This database is used for tasks such as index maintenance, database integrity checks, and query sort operations. This database's performance characteristics vary widely by application and instance, and special care must be taken to ensure that this database's performance doesn't negatively impact the performance of the task that depends on it.

Up Next

With SQL Server's I/O architecture detailed, we can now move on to discussing how the two layers—SQL Server and storage—complement (and potentially contradict) each other in the enterprise.

CHAPTER 3

SQL Server Storage Considerations

Knowing how dependent SQL Server is on storage performance, certain architectural considerations must be reviewed and analyzed (especially by any organization with enterprise data needs) to determine the impact that storage performance and SQL Server demands will have on the application.

Performance Metrics

First, let's define how to objectively demonstrate the storage performance. These metrics will help you understand the steady-state performance of the storage, and will help you track and trend storage performance to determine if it's running as expected.

The metrics you should care about:

- Latency
- IOPS
- Throughput

Latency

The amount of time taken for the round trip of one storage request from the operating system layer to the storage and back is called *latency*.

Workload Separation

For all three of these metrics, you should separate read and write operations, because the performance characteristics of these two operations can be very different.

On most SANs it's measured in milliseconds. SSD and NVMe SANs can have performance so strong that the performance can be much lower than one millisecond, usually into the microsecond range.

For storage performance, lower latency is better, and the lower the latency, the faster the SQL Server can complete the I/O request.

For certain SQL Server operations, such as an Availability Group synchronous replica commit, the transaction is committed on the secondary synchronous replicas and then the local primary replica. In these cases, the commit time to storage has a much larger impact on the performance of the operation.

IOPS

IOPS, as stated before, is the number of outstanding requests from the servers that the storage can handle at any given time.

The combination of interconnect speed, storage controller performance, cache memory, and disk configuration help to dictate the maximum number of IOPS that the SAN can deliver.

Higher available peak IOPS is critical to maintaining the SQL Server's peak performance. For highly concurrent and active SQL Servers, IOPS values and the latency per I/O can determine the scalability and high-end performance of the entire application stack.

Keep in mind that even if the storage can deliver an exceptionally high number of IOPS, the application might not demand this from the storage layer. Some SQL Server workloads do not have a workload high enough to push the IOPS demand very high.

However, even if the demand in IOPS is low, low storage latency will help improve the performance of the database requests.

Interconnect Bottlenecks

A key point with throughput is that the interconnect from the storage to the server is rated in *maximum theoretical throughput*, or how fast the end-to-end connection can move data. Even if everything is absolutely perfect in the environment, with no overhead, you will not be able to exceed this value of performance on a single interconnect path.

For example, if the storage is connected via a single 8 Gb/s fiber optics (common in data centers), every 8 bits of data gets encoded to a 10-bit payload. The other 2 bits are for data integrity.

Eight gigabits per second, converted to megabytes per second, translates to a theoretical top throughput of 850 MB/s down that individual path.

No interconnect topology is 100% efficient, and overhead comes in the form of switch hops, broadcast traffic, or multipathing overhead, so the expected maximum actual throughput of one-way traffic on this path would be in the neighborhood of 780MB/s.

This speed might be artificially limiting the peak performance of the storage. Additional interconnects with multipathing can (and should) be configured to increase the total overall throughput.

Throughput

Throughput is simply the server's formatted disk block size multiplied by the IOPS, usually measured in megabytes per second (MB/s). It's regarded as a trusted measurement of performance. However, while throughput is important, latency and IOPS are much more important to the overall system performance than MB/s.

Infrastructure Layer Metrics

When reviewing performance metrics, make sure that you sample and review the performance metrics from each layer in the system stack.

The performance metrics from each of the following items should be overlaid to determine if the end-to-end performance is meeting expectations.

- SAN controller host bus adapters/network interface cards (HBAs/NICs) throughput, by port
- SAN controller CPU performance
- SAN controller cache read and write performance and ratios
- Hypervisor (if applicable) HBA/NIC throughput, by port
- Hypervisor (if applicable) SAN LUN or datastore throughput, latency and IOPS
- VM disk latency, IOPS, and throughput, split by virtual disk
- Windows Server logical disk latency, IOPS, and throughput metrics, by drive or mount point
- SQL Server disk stall metrics, by database file

Controller Bottlenecks

A key factor affecting performance of data reducing SANs is the architecture of the SAN controllers. For SSD and NVMe-based storage, controller performance is the limiting factor. The flash memory cells are faster than the controller itself, so the controller must be able to handle any operation with minimal overhead to retain peak performance of the SAN.

SSD and NVMe-based SANs contain multi-core CPUs and a significant amount of memory for metadata management and read caching. But the core controller platform is still subject to the demands of the connected servers, and the type of configuration and demand could negatively impact the overall performance of the SAN if the controller architecture is not sufficient to accommodate the workload.

An example of this performance penalty is with data-reducing storage. Many SANs claim data-reducing functionality, such as in-line deduplication and/or compression, but beware: some contain controller hardware that's underpowered for these operations. As a result, the controllers are maxed out with processing while the storage is relatively idle, and the resulting delays cause a storage latency penalty for the connected servers.

Maximums vs. Steady State

An important distinction must be drawn between the maximum performance an individual infrastructure component can deliver vs. the ongoing "normal" operational range of these components.

Think of it like this. The next time you're in your car, take a look at how high the speedometer goes. When was the last time you maxed out the

Storage Maximums Testing

Microsoft's *diskspd* utility can be used to stress test the storage underneath your SQL Server. You can configure numerous parameters to simulate any type and intensity of workload needed.

While not a true test of a SQL Server workload—the workload can impose numerous storage patterns at the same time—this utility can be used to simulate a single particular workload type per test. Repeated tests with different parameters can help determine the storage response times for the various types of workloads coming from the SQL Server.

To better simulate a SQL Server workload, what better way than with the actual workload itself? SQL Server Distributed Replay, a feature included starting with the 2012 release, can be used in conjunction with a production database backup and trace to process and replay the recorded production workload against a copy of the database.

speedometer? Can your car actually reach the maximum speed listed on the speedometer?

Storage performance is the same way. Under normal circumstances, the storage IOPS and throughput demands are usually well under the maximum capabilities of the storage. However, occasionally, some SQL Servers do demand that peak performance.

Cache vs. Primary Storage

The SAN contains both cache and primary storage drives as part of the basic configuration. Cache is considered transient buffering, for both frequently accessed storage blocks and for data write operations.

Cache is usually many times faster than the permanent storage layer. The goal is to improve the performance of the SAN while reducing the latencies in reading data from, or writing data to, the persistent storage medium.

Legacy SAN cache uses small amounts of battery-backed memory to buffer these requests; modern SAN technologies use a combination of large amounts of memory and SSD to increase cache performance and capacity, which in turn improves overall SAN performance.

When data is written to the SAN by a server, the controller accepts the request and writes this data into the controller cache (**Figure 3-1**). The SAN then returns an acknowledgement that the write completed successfully (**Figure 3-2**).

At this point, the SAN copies the newly-written data to the internal permanent primary storage layer. The block of data will eventually be written to the primary storage layer (**Figure 3-3**).

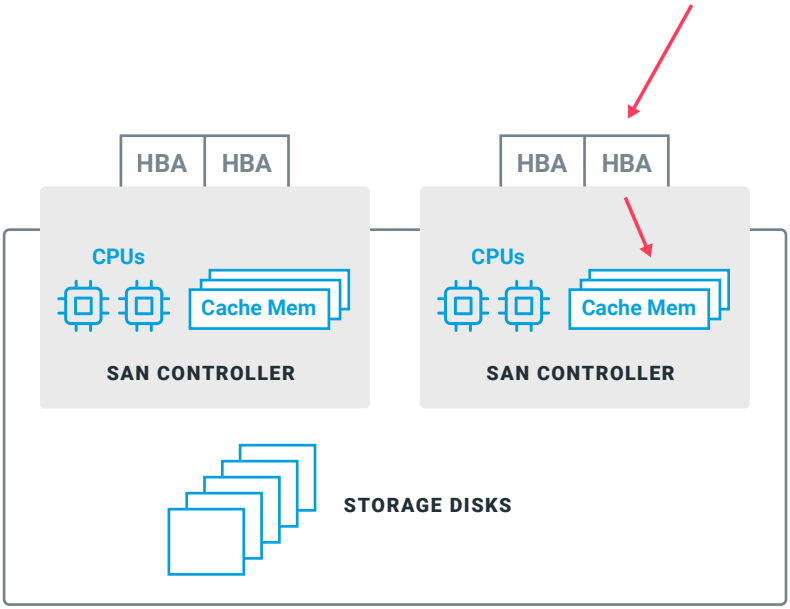


Figure 3-1: A SAN controller receives a write.

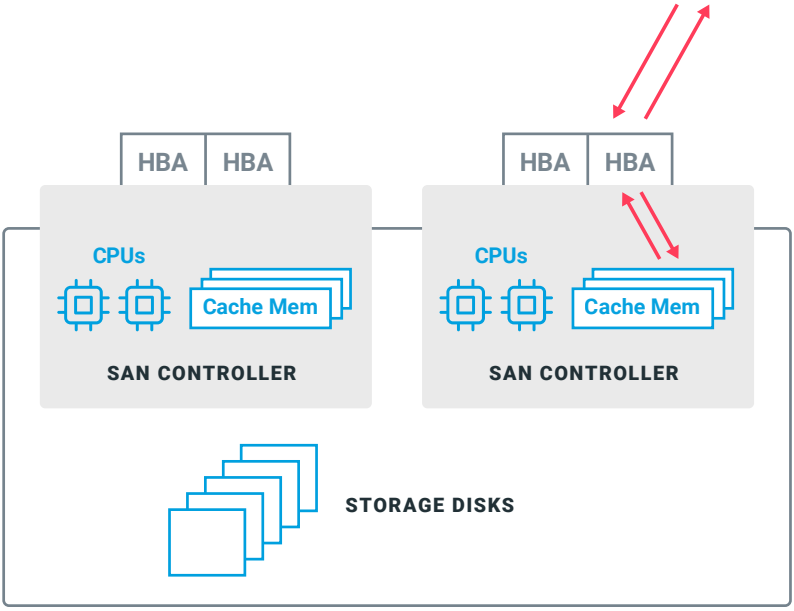


Figure 3-2: The SAN controller acknowledges the write.

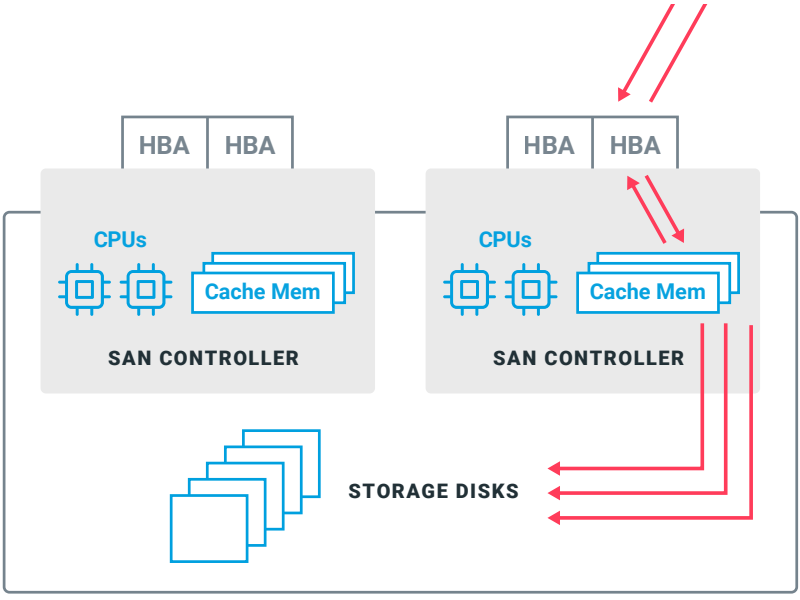


Figure 3-3: The SAN controller flushes the write to disk.

If the SAN has a large amount of cache, it can retain a copy of it in this layer in case this data is requested in a read operation in the near future.

Large amounts of read cache can improve overall SAN performance if the workload is largely made up of read operations. Database operations are more than 80% read in nature, and large amounts of SAN cache usually improves the overall performance of the database.

Disk Configurations

Modern SANs come in shapes and sizes big and small, and the performance capabilities range from slow to lightning fast.

Traditional

Traditional SANs (an example is shown in **Figure 3-4**) contain a number of identical spindle-based primary storage disks and a small amount of cache memory. These arrays were historically built for redundancy and resiliency. Improved speed could be introduced, with a greater number of spindles and larger amounts of cache memory.

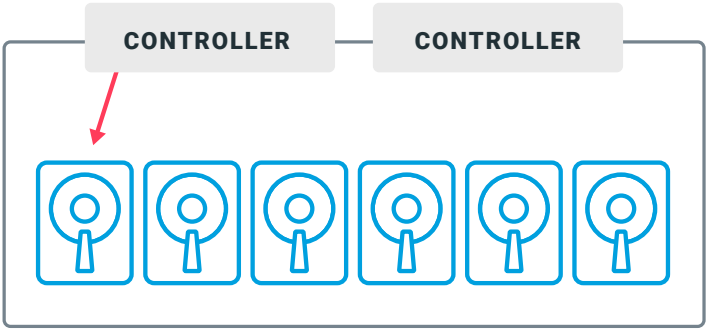


Figure 3-4: A traditional storage array.

Tiered

Tiered SANs (**Figure 3-5**) use multiple groups of disks, each with different performance characteristics. Data is usually placed on the fastest tier of disk first. If the data is untouched for some length of time, the system can relocate the block to a slower (and less expensive) tier of storage during a periodically scheduled maintenance process.

Over time, commonly-accessed data is kept at the fastest layer, and archival data sinks to the bottom of the performance tier. While cost effective, the data movement process can tax the system. Accessing infrequently used data can cause the operation to run very slowly because of the disk speed where that data is stored.

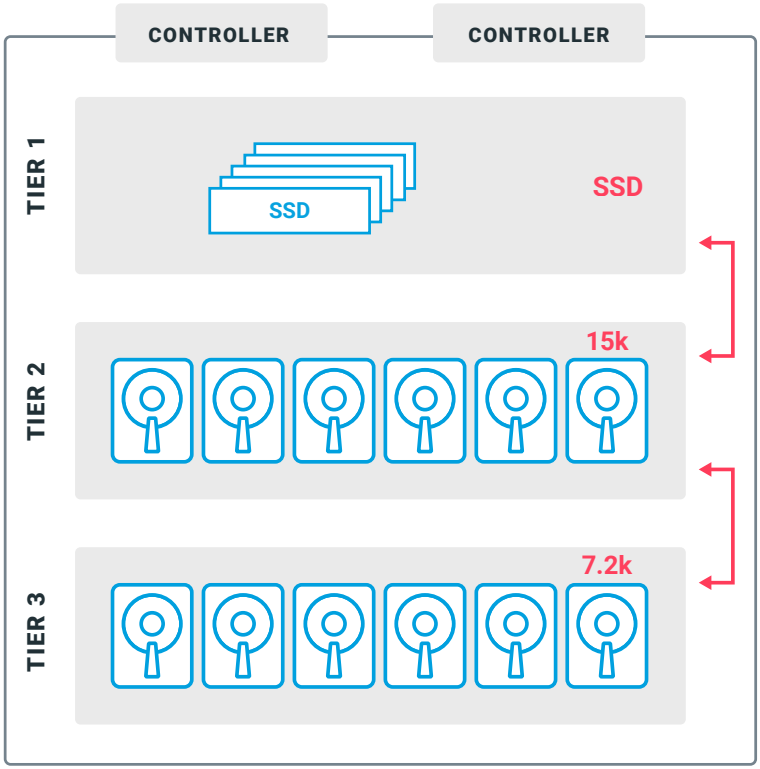


Figure 3-5: A tiered storage array.

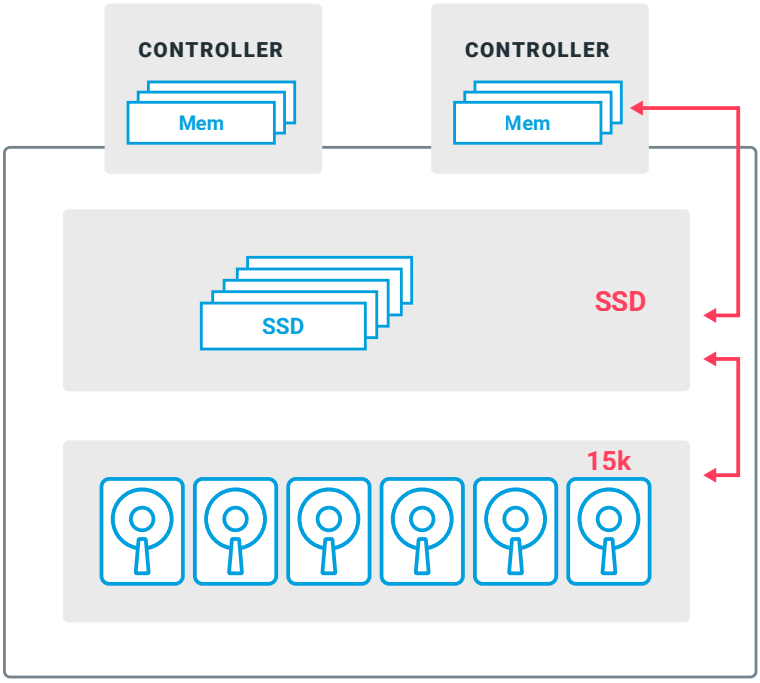


Figure 3-6: A hybrid storage array.

Hybrid

The tiered SAN model, with the manual movement of data tied to a process that only occurs infrequently during a maintenance operation, has given rise to hybrid SANs.

Hybrid SANs (shown in **Figure 3-6**) perform a similar type of hot and cold data relocation process to tiered SANs, but do so automatically and around the clock, instead of during a particular maintenance window. This improves the overall performance of the SAN.

Hybrid SANs usually also contain large amounts of cache memory, to improve the performance of read operations. If an active block of data is constantly re-read from the SAN, the SAN will keep this block in

RAM to deliver the fastest possible performance for this operation, as RAM is many times faster than SSDs or spindle disks.

Hybrid SANs also ensure that all write operations are performed on the SSD layer, so that the operation is performed as fast as possible while maintaining the critical integrity of the data being written to disk.

Common configurations can include a two-layer solution with a tier of solid-state drive (SSD) disks for very fast write operations, and a tier of SAS spindle-based disks for longer-term data placement.

All-Flash

For the most demanding of workloads, all-SSD (all-flash) storage can be leveraged to provide very impressive performance of the shared storage models (**Figure 3-7**). While more expensive than traditional or hybrid SANs, all-flash SANs provide significant performance compared to spindle-based SANs.

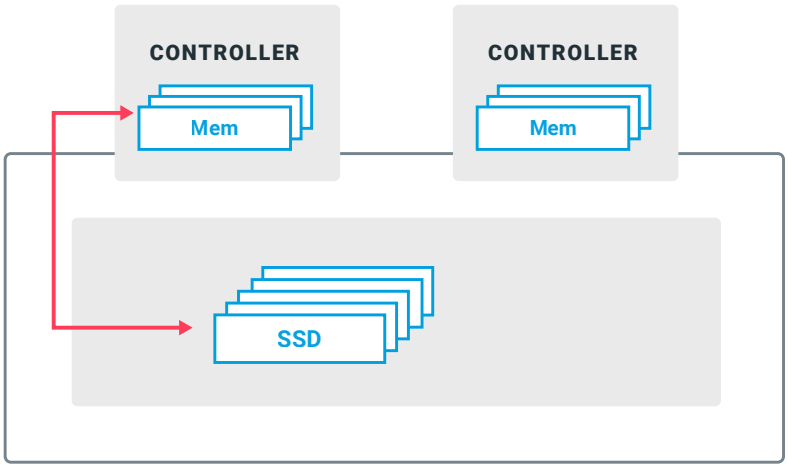


Figure 3-7: An all-flash storage array.

Even with the impressive speed of the all-flash SSD layer, large amounts of cache memory are leveraged in each controller to improve the overall performance of the array that much more.

NVMe

Recently, NVMe was introduced into the storage arena. Traditional SSDs have inherent bottlenecks in their architecture, as their core underpinnings are tied to the SCSI storage protocol.

The SCSI storage protocol has a single I/O request queue, and contains a maximum of 256 commands. The protocol rarely emerged as a performance drain in the days when the underlying spindle-based storage could not keep up.

However, as the underlying storage media increased its speed with SSD, this core architectural limitation became the bottleneck to performance of a single SSD disk.

The NVMe standard was developed to overcome this shortcoming. It's an improved standard for nonvolatile memory access, as well as a new instruction set and communications protocol designed to leverage the parallel access of this new form of storage.

As a result, instead of a single queue with a maximum of 256 commands, the NVMe standard contains 65,535 queues, and each queue has a maximum queue depth of 65,535 commands. These storage devices connect directly to the processor via the PCIe interface, which provides significantly greater throughput and reduced latency properties compared to legacy protocols.

The result is that NVMe-based SANs provide much lower reductions in storage latency, improved performance for random I/O patterns (such as dense virtualized environments and relational databases), and continued gains in available throughput.

Efficiencies and Data Reduction

Newer SAN architectures can incorporate features that allow greater efficiencies in the data storage so that more data can be squeezed onto the SAN.

Compression can be used to reduce the footprint of the data stored on disk, and deduplication can be leveraged to make the disk storage and retrieval processes even more efficient. (The catalog of where the data is stored on the array can also be more efficiently managed, improving the performance as well.)

Compression

A block of data might not contain all random and unique data. Usually, empty space and repeating patterns exist in portions of the data. Compression algorithms can be used to compress the block of data as it is passed into the SAN, and decompressed on its way back out. All of these operations occur transparently to the servers reading and writing to the SAN.

This compression algorithm must be efficient, and the storage controller must have enough processing power to handle these requests fast enough, or the overall read and write process can slow down the performance of the SAN. The result is lower latency, IOPS, and throughput performance metrics from the SAN, which, in turn, can slow down the performance of the business applications.

Deduplication

The blocks of data written to disk can also be redundant in nature. Databases from a production server can be backed up and restored onto a pre-production server for testing. These blocks of data are identical in nature, and just take up extra space on disk. As a result, some SANs

have deduplication algorithms built into the units to reduce the data written to disk.

If a block of data is to be written, and it is identical to another block, a pointer is written to the original block instead of writing the identical block a second time. The net result is additional space savings on disk.

Most blocks of data are compressible, and many environments have large quantities of duplicate data. The space savings can be upward of 40% to 70% on disk.

As with compression, the SAN controllers must be fast enough to perform this deduplication without imposing a performance penalty on the I/O operation.

Data Savings

Either compression or deduplication can reduce the storage consumed on disk, and when combined, can contribute to impressive savings on disk. The net result is that the amount of data written to the storage is smaller than the amount that the server "sees."

Most blocks of data are compressible, and many environments have large quantities of duplicate data. The space savings can be upward of 40% to 70% on disk. The performance of the SAN can improve as well, as the SAN must retrieve fewer blocks of disk from primary storage to fulfill a request.

For example, 20 TB of enterprise data could be placed on a SAN with only 12 TB of actual usable storage, and the compression and deduplication could provide additional storage space to be used by the business.

Layers and Roles

Data reduction is usually more efficiently handled by the SAN than within the server. Communicate with your SQL Server DBAs about the use of compression and deduplication in your environment.

Some DBAs have data compression features within the database at their disposal, but beware: Having data compression enabled inside the database through the use of features such as table-level page or row compression, or highly compressed objects such as columnstore indexing, can greatly reduce the effectiveness of the SAN's data reduction algorithms and reduce SAN performance.

Note, however, that there are still times that use of compression in the database layer results in benefits that outweigh the loss of SAN compression. There is no one-size-fits-all rule!

Metadata Management

On some SANs, the catalog of the location of each block of data—the metadata—is commingled on the data disks. Every time a block of data is fetched from the SAN, this metadata must be referenced to determine where to fetch that block of data from, which introduces additional overhead on the array.

On other SANs, this metadata is handled and stored separately from the data. Tegile, for example, stores metadata on SSD disks and mirrors it in memory, which ensures that the lookup of the data blocks is performed as fast as possible. Any changes to the metadata are written to the SSD metadata disks. The operational speed of managing the metadata separately from the data layer improves the efficiency of the array, which improves the overall performance.

Up Next

Storage architecture matters with SQL Server, and certain SAN features can improve the performance of the database while reducing the space footprint at the same time.

Let's move on to learning different business continuity strategies for both SQL Server and storage.

CHAPTER 4

Business Continuity & Disaster Recovery

The data in your database must be protected from data loss at all costs. Data loss is unacceptable these days, and the consequences to the business can be dire.

Both the SAN array and the database engine provide excellent methods for protecting your data and replicating it to a different array. Sometimes these features are complementary, but other times one feature can directly compete with another. Which is the right strategy for your organization?

Business Continuity and Disaster Recovery Via Storage

SANs provide a number of different methods to protect data against human error and technology failures.

RAID

The SAN itself provides some internal protections against data loss. RAID is a key component of all SAN disk configurations. It allows for the data to span more than one internal disk. That way, if a drive were to fail, no data is lost, and the array can (eventually) rebuild the data that was on the failed drive.

Definitions

First and foremost, let's define some terms about key aspects of business continuity. These items must be identified by the business before any technical solution can be implemented.

Recovery Point Objective (RPO)

Recovery Point Objective (RPO) is the block of data, represented by time, that the business will tolerate losing. For example, if a business sets four hours as the RPO, then—at a minimum—the data must be backed up and successfully replicated every four hours. Reducing the RPO means that data must be successfully backed up more frequently.

RPO strategies become more complex as the RPO decreases.

Recovery Time Objective (RTO)

If you're willing to lose a certain amount of data, the question then becomes how long you're willing to wait until the rest of the data is restored and accessible. The *Recovery Time Objective* (RTO) is the maximum length of time that the systems can stay offline for a given situation.

As with RPO, the strategies get more complex as the RTO gets closer to zero.

RAID types like RAID 10 and 5 have been used for years by storage admins to protect the data, but these days the SAN usually has a preferred RAID configuration for its architecture, and you might not want to change it.

Snapshots

SANs also have features that help provide recovery points that can insulate the business against software and user error. LUN-level snapshots are a point-in-time recovery point on a LUN, and can be taken without downtime to the application then reverted back as needed in the event of a problem.

These snapshots can be scheduled to run at routine intervals across the business day, and will clean up after themselves after a fixed window of time has passed.

LUN-level snapshots are a point-in-time recovery point on a LUN, and can be taken without downtime to the application then reverted back as needed in the event of a problem.

Snapshots are very useful. Consider a real-world example: A snapshot is taken just before an application upgrade begins. Let's say that during the upgrade, the upgrade process fails, but only after an entire database table has been updated. The only option is to undo everything that changed all the way back to the point just before the upgrade.

Without a LUN snapshot, the application owner would have to roll back every change that the failed upgrade process did not clean up, and the DBA would have to restore a database back to the point before the upgrade. Either of these steps could take hours or even longer.

The LUN where all of this data resides could be rolled back to that point in time in just seconds, undoing any change with an automated process that is sure to save the business hours. The environment is then reset so that the business can resume operations while they reset and figure out why the upgrade failed.

Replication

SAN-level replication is when a SAN LUN is configured to replicate its contents to another LUN on another SAN (either in the primary data center or to a remote device). This process also occurs at the LUN. It can be configured to run synchronously or asynchronously.

Asynchronous replication can be configured for either a constant stream of changed blocks, or to run periodically on a schedule. Usually, asynchronous LUN-level replication is configured for a window of 15 - 30 minutes.

This replication frequency schedule should be dictated by the RPO. If the LUN-level replication features on your SAN are unable to meet the RPO, it's time to turn to SQL Server for assistance.

Types of Replication

Two types of data replication architectures exist for moving data between SANs: synchronous and asynchronous.

Synchronous replication means that for every block of data written to the SAN, the block gets mirrored to another SAN and successfully written to disk before the I/O operation completes successfully.

This type of operation is usually used within a single data center, because the slower speeds between data centers can reduce the performance of an I/O operation's round trip.

Asynchronous replication means that any data change to the primary array gets queued up and sent over to the other array as fast as possible.

Care must be taken with LUN-level replication when used with SQL Server. If the various files that make up a SQL Server database are on different LUNs, all of these LUNs should be replicated in the same consistency group. If not, the SQL Server database will probably fail to come online in the event of a failover, because the different pieces are out of sync.

Business Continuity and Disaster Recovery Via SQL Server

Out of the box, SQL Server has a number of strong business continuity features that can be configured in an infinite number of ways to meet your needs.

Database and Transaction Log Backups

Depending on the RPO and RTO dictated by the business, extending SQL Server to meet the business continuity needs could be as simple as periodic backups of databases copied to a disaster recovery (DR) site.

Log Shipping

If the RPO window is very small, database transaction log shipping can be utilized. Every change that occurs in the database is logged to a transaction log, and this file can be automatically backed up, the file copied, and applied to a standby database on a second database instance at another site.

Log shipping is extremely robust, and has been used by DBAs for decades to provide a simple and straightforward business continuity strategy.

Availability Groups

SQL Server's flagship availability feature, "Always On" Availability Groups (AGs), can replicate data for both high availability (HA) and DR purposes, and can do so at up to eight additional copies of the data (as of the SQL Server 2017 release).

Synchronous or asynchronous data replications can be established to keep the RPO down to as low as seconds, based on the rate of data change and the bandwidth between the two SQL Servers. (This feature is available only in SQL Server Enterprise edition. Basic AGs are available in other editions. They're limited, but they work for some workloads.)

Competing or Complementary?

Knowing the features available with both the storage layer and SQL Server is only the beginning of the architectural decisions an organization must face when designing a solid business continuity plan for the data.

Both replication strategies present compelling arguments and solid features for use by a business to protect their data. Let your RPOs, RTOs, and in-house skill sets determine the appropriate technology for each application stack.

These features can be used to complement each other, such as SAN LUN-level replication to a secondary array located in the public cloud. The VMs can be replicated between sites. Database data can be moved between servers, data centers, or entire clouds, completely transparent to the application. Even the file-system data from within the VMs can be replicated independently of other solutions. Use the combination of features that best suits your organizational needs for business continuity.

Be careful about the technologies in use. For example, if LUN-level replication is in use, and a DBA enables database log shipping to reduce

the failover time, the changed database data is being moved to the secondary site twice. This configuration consumes twice the WAN bandwidth, and can clog the connection, which will result in both data streams slowing down.

One consideration when architecting business continuity for data platforms is to determine which group (or groups) is responsible for the DR process. In some organizations, the infrastructure team is solely responsible for the DR architecture, and the SQL Server environments are treated like every other server. Data replication, failover and failback processes are identical with the other servers to keep the process as streamlined and as simple as possible.

Other environments let the different applications select the business continuity strategy that best suits the RPOs, RTOs, and skillsets of the administrators. If the RPO for the application-layer is good enough for the application itself but not the underlying data, a combination of strategies can meet the RPO and RTO for the entire application stack while keeping the architecture as simple as possible.

Up Next

The options and strategies for protecting the database are endless, so choose wisely. Next, we'll discuss SQL Server implementation features and how they relate to the storage architecture.

CHAPTER 5

SQL Server Considerations

Successfully managing the intersection of SQL Server and storage technologies is crucial for any organization. Knowledge of how SQL Server uses I/O for performance can help a storage administrator design the infrastructure so it improves the efficiency of the entire stack.

In this chapter, we'll explore ways in which storage can be leveraged to improve the performance or operational overhead of SQL Server.

SQL Server vs. Array Features

Sometimes SQL Server features can be quite complementary to the SAN's set of features. Other times, SQL Server features can wreck a SAN's performance.

Compression

SQL Server DBAs can leverage table-level row or page compression to reduce the data footprint stored on disk. This compression is even maintained when the data is loaded into memory for processing, increasing the amount of data that the SQL Server buffer pool memory can hold. The SQL Server CPUs are marginally busier, but the reduced read I/O can improve the performance of the SQL Server database.

However, if compression and/or deduplication is enabled underneath the SQL Server data, compressed data is not further compressible. All it does is create a much greater workload for the data savings architecture on the SAN.

Layers and Roles

Be sure to discuss with all parties involved where compression and deduplication is more effective: at the SQL Server or at the SAN. Usually, the SAN is more efficient at managing the deduplication, and can achieve greater data savings.

The net results are greater SAN controller CPU consumption, higher latencies for the SQL Server storage, and much lower data savings rates on the physical storage.

Transparent Data Encryption

SQL Server's Transparent Data Encryption (TDE) allows administrators to configure a very robust database-level encryption that encrypts the database on disk. Security is absolutely vital, and encrypting data at the database level is required in some organizations.

As with compression inside the SQL Server database, TDE can also cause the data savings percentages on disk to plummet and the workload on the SAN to rise.

Most SANs contain a feature to enable encryption of the storage layer, thus enabling encryption of data at rest. If the organization security policy allows, consider encrypting the data on disk rather than inside the SQL Server engine.

The security at the SAN layer could be enough to satisfy organizational requirements, and the data savings will allow much more data to be placed on the SAN efficiently.

Partitioning and File Groups

Some SQL Server tables can be quite large, with billions of records stored. Queries can slam these tables hard, causing full table scans to funnel down one table and into one path to the storage.

SQL Server can leverage table-level partitioning to reduce the amount of data that a query will have to scan to find the necessary records. Data can be grouped by whatever logical grouping best fits the data. A partition could be defined on a monthly basis for order history, and a partition would exist for each month that an order has been recorded (**Figure 5-1**).

The query engine can then align the I/O requests across only the partitions which contain the data that the query requests.

Commonly joined tables are normally placed in the same database filegroup and data file, which forces the storage engine to funnel the requests down one path to disk.

Why does this matter with regard to storage? The files underneath database partitions or filegroups can be placed on multiple drives, corresponding to multiple disks, SAN LUNs, active controllers, and disk pools.

Encryption Layers

Keep in mind that this LUN-level encryption is at the SAN level only. If the VM is backed up or cloned, and this secondary copy placed on unencrypted storage, the security layer is removed. Encryption at the database layer moves with the VM through the data management life cycle.

SQL Server will better parallelize the I/O operations within the SQL Server engine. The overall performance of the I/O task can be increased by spreading out the corresponding workload underneath SQL Server, reducing the impact of infrastructure or SAN controller or disk pool bottlenecks. This configuration can make the entire I/O operation more efficient (**Figure 5-2**).

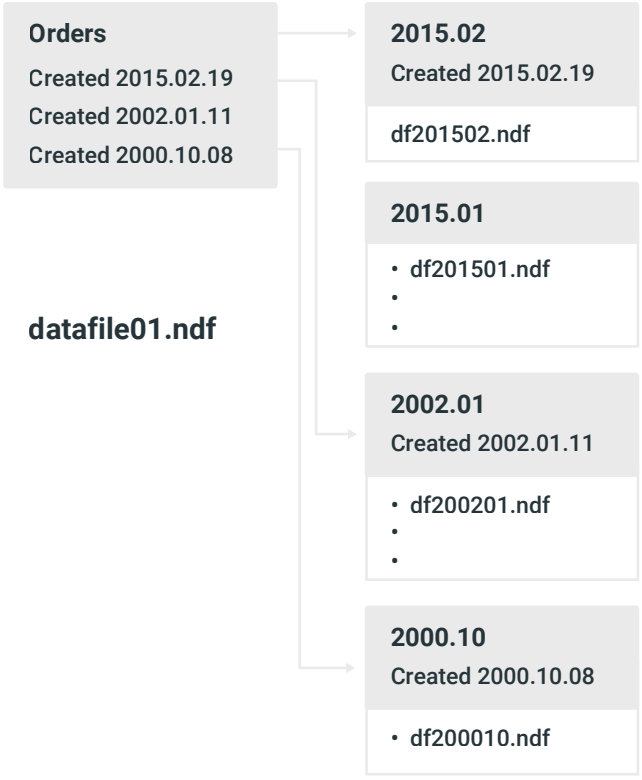


Figure 5-1: Spreading out the database workload.

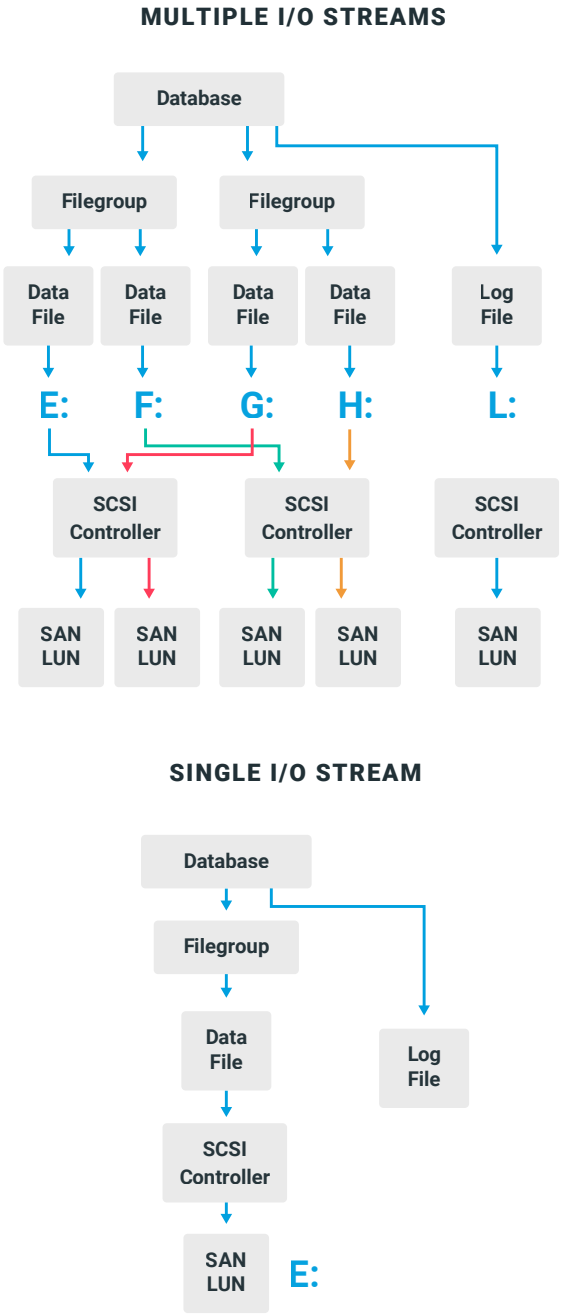


Figure 5-2: An example of a SQL Server workload spread.

Columnstore Indexes

SQL Server DBAs can also leverage a newer construct called "columnstore indexes," which can improve the performance of certain warehouse-style operations on relational database tables.

Its storage architecture is significantly compressed, even at the expense of the extra CPU cycles needed to compress the data upon a write operation. These indexes are quite useful for DBAs to leverage in their database design, and can improve certain query runtimes by orders of magnitude.

SQL Server DBAs can also leverage a newer construct called "columnstore indexes," which can improve the performance of certain warehouse-style operations on relational database tables.

Rather than discouraging DBAs from using this technology, consider placing the columnstore index itself on a different data file in a different filegroup, and ensuring that the underlying LUN configuration doesn't impose a performance penalty on its operations.

Buffer Pool Extensions

SQL Server can leverage a buffer pool extensions (BPE) file placed on a flash-housed drive as a spill location for memory pressure.

If the SQL Server is under memory pressure and a BPE file has been defined, SQL Server can leverage this file and its underlying speed to extend the assigned memory to improve performance. This file can then be placed on a flash tier of a SAN, which will boost SQL Server performance.

Your Mileage May Vary

Every organization's relationship with Microsoft licensing is different. While the advice you find here can be used to save your organization money, please verify with your Microsoft licensing representative that your licensing arrangement lets you leverage these suggestions.

SQL Server Licensing Reduction

The SQL Server licensing discussion is at the forefront of a CFO's mind during the yearly renewal cycle, since SQL Server is a very expensive application. In fact, the cost to license SQL Server could outweigh the cost of the physical server that it runs on, or even the cost of the SAN where its data is stored. Reducing the associated costs of SQL Server is of key interest to management.

Two key strategies exist to save your organization money on licensing: virtualization and flash storage.

Assuming your SQL Servers are virtualized, let's talk tech. SQL Server licensing is based on three models:

- **Server plus client access license (CAL).** Each operating system that runs SQL Server is licensed; then a CAL is purchased for every named user or device that connects to the SQL Server. Depending on the scale of your environment, this model might save you money, but it's rarely used for large-scale deployments. Server + CAL licensing is now only available on Standard Edition, and only for grandfathered organizations.

- **Per VM (by core).** Each VM must be licensed by the number of cores assigned (minimum of four cores). Standard and Enterprise edition can each be licensed in this manner.
- **Per host (by core).** All physical CPU cores must be licensed on each virtualization host where the VM could migrate to, and run from. (Only Enterprise edition can be licensed in this manner.) If the hosts are properly licensed as part of Software Assurance, your organization is free to deploy as many SQL Server VMs as can fit onto the licensed hosts.

So, for example, let's assume you have a decent number of SQL Server VMs.

Most modern virtualization hosts are quite powerful. They contain a large number of physical CPU cores and memory.

It's a good thing, because SQL Server, whether physical or virtual, needs a large amount of memory to operate efficiently. This memory is used for the *buffer pool*, which is an in-memory read cache for commonly-accessed data. Historically, this buffer pool was kept large in order to improve the performance when the storage layer performed poorly.

Virtualizing Your Databases

If your organization doesn't have your SQL Servers virtualized by now, weigh the options and consider virtualizing the databases as soon as you can. The agility and flexibility virtualization brings can save your administrators countless hours over the course of a year by easing the pain of tasks such as backups, DR, server maintenance, and capacity management.

When the administrator starts to pile SQL Server VMs onto a single host, usually the first thing that fills up is the host memory. The host CPUs are usually not under pressure at all, while the host is at the maximum number of VMs it can contain, all because the memory consumption is maxed.

Since you can deploy as many SQL Server VMs as you want on the hosts you've licensed, you can now work to reduce the number of hosts that need a license. If you can reduce just one host from the license footprint, the amount of savings for the organization could be as high as $125,000.

That's serious money.

Enter all-flash SANs.

All-flash storage is extremely fast, and the latencies very low. In some cases, SQL Server might not need to keep as much of this commonly-accessed data in memory because it can fetch it from the storage layer *fast enough* to maintain performance.

This puts you in a position to reduce the memory assigned to the VM. If you can reduce the memory and maintain performance, the number of VMs on that host can go up. If this number can go up enough, you might be able to squeeze all of the SQL Server VMs comfortably on one fewer physical host.

Since you can deploy as many SQL Server VMs as you want on the hosts you've licensed, you can now work to reduce the number of hosts that need a license. If you can reduce just one host from the license footprint, the amount of savings for the organization could be as high as $125,000.

That's serious money.

CPU Performance and Licensing

In addition to performance gains from storage, the act of upgrading the SQL Server to a newer platform can improve performance.

In addition, the CPU selection in the new platform can boost overall performance of the SQL Server on a per-core basis, which could lead to an assigned CPU core reduction. That can contribute to an overall licensing reduction, saving even more.

Geekbench is a benchmarking tool that provides a pair of performance metrics: a per-CPU-core score and a multi-core score for all available CPUs. Its results most closely align with the relative performance differences seen with CPU upgrades underneath modern SQL Servers.

Geekbench has a publicly accessible listing of the scores for given CPU types submitted by users of the software, available at browser.geekbench.com. The single-core score is preferential, as the number of cores can vary by testing server if the platform is virtual.

For example, a search for "Xeon 6146 Windows" (as the OS type can impact the resultant scores) returns an average of 4889 for the single-core score (as of the time of this writing). Compare this number to a very common and fast processor for its day, the Intel Xeon E5-2667 v3; the single-core score average of 3552 yields an approximate performance gain of more than 37% for the newer CPU.

Based on the workload, the additional compute performance of the CPU cores on the newer CPU could result in the SQL Server needing fewer CPUs. As a result, the CPU licensing footprint could decrease.

Monitoring, Management, and Support

Monitoring

Monitoring storage performance must come from many levels. Monitoring utilities that collect and analyze data across platforms is

key to understanding the cause-and-effect relationship that one infrastructure component has with other components.

Education is also important. Personnel from one team might see performance metrics from another infrastructure layer and not understand how to interpret its true meaning. This can lead to a proactive warning being ignored and the situation turning into a reactive support call in the middle of the night because of a critical system failure.

Monitoring storage performance must come from many levels. Monitoring utilities that collect and analyze data across platforms is key to understanding the cause-and-effect relationship that one infrastructure component has with other components.

An exceptionally common example of this monitoring difference comes into play if a SQL Server DBA sees a number of "long I/O alerts" in the SQL Server error log. A long I/O alert is triggered within the database engine if a single I/O operation takes longer than 15 seconds.

Fifteen seconds is an astronomically long time in the database world, and these warnings should be taken seriously. However, if the storage administrator reviews the storage performance metrics from the SAN at the same point in time, they might see that the storage is simultaneously showing sub-millisecond performance.

Both points of view are accurate, which is why knowledge about all the different layers and queues between the two metrics makes a huge difference. Assuming a hypervisor exists, at least five different storage queues are present above the SAN and below the SQL Server database file, and each and every one could have a misconfiguration that contributes to storage queueing with associated elevations in latency.

Management

Managing SQL Server in an enterprise takes a conscious effort to get input from those involved with all layers of the infrastructure and application stack, including:

- SAN
- Interconnects
- Virtualization
- Server/OS
- SQL Server
- Application owners

Working together to set performance SLAs, availability requirements, RPO and RTOs, and support statements is critical to properly managing these platforms. Without effective communication between the teams, any performance or availability anomaly will result in endless finger-pointing and the blame game; this is obviously counter-productive for the business.

When each team understands the high-level details about the other team's responsibilities and platforms, the technical challenges are much more likely to be resolved quickly and easily.

Support

Validating your SQL Server configuration with your storage vendor's official SQL Server reference architecture is also recommended. The SAN vendor might have special settings or best practices you should follow from the start; pay attention, since they can help improve the performance and availability of the SQL Server on their storage platform.

Up Next

SQL Server is a lot more complex than many administrators realize; but with a solid understanding of its features and capabilities, architects can improve the power of the entire stack and the speed at which the business can operate, all while reducing the operational costs of the platform. Next we'll discuss specific recommendations for the SQL Server and storage layers that can squeeze the most performance from your environment.

CHAPTER 6

Storage Best Practices

Now that you've seen how important storage performance is to SQL Server performance, the natural question is how to get that performance.

You first need to see if something else in the infrastructure is artificially slowing down the SAN's performance. Once you've rectified any infrastructure bottlenecks on the way to the storage, it's time to validate the SAN configuration itself.

Certain settings and configurations can be tweaked to improve performance at each layer of the infrastructure; depending on the type of SAN in place, these details can make your SQL Servers all-star performers.

Best Practices Start Early

Fixing storage misconfigurations and pain points starts as soon as you get the array in the rack. Some of the misconfigurations and challenges are infinitely easier to detect and fix before you start putting your data onto the storage for production purposes.

Thoroughly test any new storage purchase to ensure you're receiving the expected performance from the device *before* it goes into production.

Hardware and Bottleneck Detection

Bottlenecks inside the infrastructure can be hard to diagnose. One large pain point between the data and the application can cause the entire system to slow to a crawl.

Looking at one layer of the system might not reveal the root cause, because the symptoms of the problem can change from each perspective. This is despite the fact that the underlying problem could be the same. Misconfigurations can, and do, cause these sort of problems in the infrastructure.

Many go largely undiagnosed because the teams who manage the layers don't recognize that the symptoms other teams report may be initiated within their own layer.

This is why it's crucial to take a high-level holistic view of the environment and start to drill into each layer, putting performance metrics at each point along the way.

Storage Bottlenecks

Figure 6-1 shows some of the potential storage bottleneck points along the path from disk to the SQL Server.

These bottlenecks could be due to a number of things:

- The disk pool or RAID configuration in the SAN could simply be slower than the SQL Server workload demands.
- The SAN controller could have too small a cache to adequately buffer the workload.
- The controller's port to the storage interconnect could also be misconfigured or at its peak capacity.

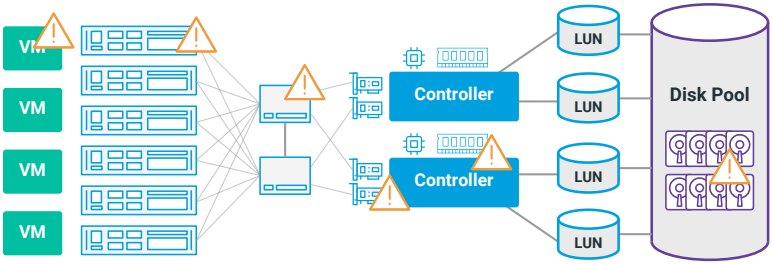


Figure 6-1: Storage bottlenecks can occur at any of these points.

- The interconnect switch or switches could be misconfigured or at peak capacity, or even slower than the rest of the interconnects. This effectively slows down the entire path.
- The interconnect port on the physical server could be at peak capacity or have a misconfiguration.

Additionally, if a hypervisor is being used, several layers could be misconfigured or mismanaged, and can slow down the I/O request even more.

Operating System Bottlenecks

Even more bottlenecks can exist inside the OS. They can sit at the disk controller level (physical or virtual) within Windows Server, inside the SQL Server engine, or inside the database data files (**Figure 6-2**.)

Ordinarily, the end result of this series of challenges is increased latency within SQL Server and Windows Server, compared to the SAN. So the storage administrators see very good performance and low latencies, but the DBAs see very poor performance and very high latencies.

The strategy to resolve this challenge is with performance data. At each and every stop between the two sides, measure the appropriate performance metrics and start to put the same type of metric in place end-to-end. If the numbers differ wildly between two points, stop and

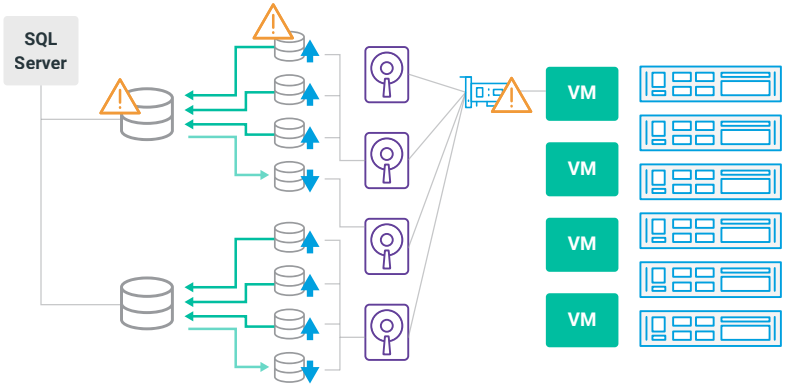


Figure 6-2: Potential points for operating system bottlenecks.

drill into that portion of the topology to determine where there's room for improvement.

For example, if the SAN reports 1.5ms response time for read operations on a given LUN, and the hypervisor shows about the same value to the same LUN at the same point in time, the interconnect layer is probably in good order.

However, if the Windows layer inside the VM that sits on the LUN reports a read latency of 25ms at the same point in time, check the hypervisor and Windows-level disk configurations.

Objective numbers don't lie, and they can help resolve disagreements between teams. The goal is to have the best possible performance of the application (and all layers beneath it) to provide the business with the best possible tools to excel in the face of stiff competition.

Work together, put the numbers to each layer, and let the performance values drive the investigation.

Block Size

Block size is an overlooked portion of tuning the environment to improve the storage performance. Block size is the group of contiguous space used to manage data placement on disk. Each layer of storage, from Windows Server, to the SAN LUN architecture, to the SAN disk configuration, manages data placement in these blocks.

For example, Windows Server's default block size is 4KB. Using round numbers for the example, if the C: drive is 100GB in size, it's broken into roughly 26 million logical units of 4KB per unit.

Windows Server 2008 and later also use a 1MB partition offset so that the partition starts far enough into the disk to avoid any strange block sizes or offsets from the storage layer.

SQL Server on Linux® Block Size

Microsoft recommends a 64KB NTFS allocation unit block size for Windows Server logical drives. But with the recent inception of SQL Server on Linux, block size recommendations for the supported Linux file systems (EXT4 and XFS) are less concrete.

Changing the EXT4 block size is extraordinarily difficult because of the page size in the kernel of 4096 bytes. XFS, on the other hand, can support a larger block size. Extrapolated from the following link, Microsoft unofficially recommends using a 2MB block with the XFS file system for SQL Server on Linux deployments. [1]

[1] https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-configure-pmem

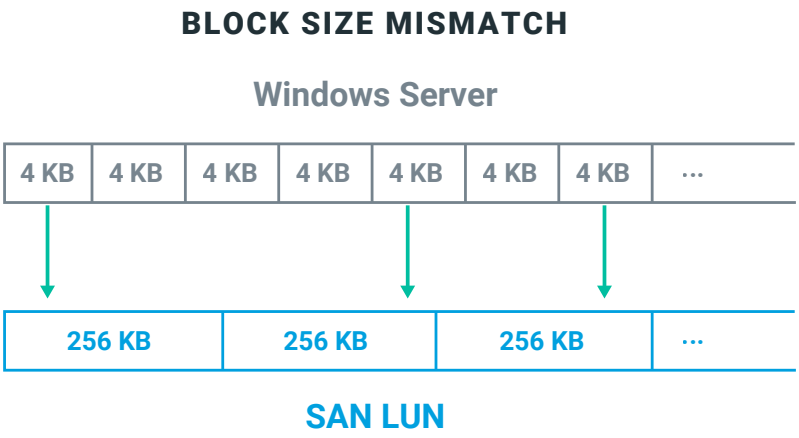


Figure 6-3: A typical block size mismatch.

The SAN disk and LUN configuration can also be aligned on a block size. But what if these block sizes are different within Windows from the SAN LUN configuration?

Figure 6-3 shows a Windows Server NTFS configured for a 4KB block size. The SAN LUN is configured for a 256KB block size. If the workload on the Windows layer is random, a single 4KB block read could force the SAN to read 256KB of data just to fetch the 4KB and return it back to Windows. Now repeat this millions of times over the course of a day for a normal server. The storage layer might be reading much more data from disk than is necessary, and the result is slower overall performance for the workload.

As a result, consult with your storage vendor's validated SQL Server reference architecture for the recommended LUN block size. Configure the SAN LUN per their recommendations, then test to verify performance.

For most workloads and environments, a 64KB block alignment strategy is ideal within the SQL Server and Windows layers. Occasionally, an alternate block size might provide slight performance improvements to SQL Server, but extensive testing should be performed to come to that determination.

Server-Based Storage Connectivity

Whether a SQL Server is physical or virtual, the storage must be presented to the operating system. Many methods can be used to perform this task, and they're not all created equal.

For physical servers with locally-attached storage, a storage controller is usually used to manage RAID, write caching operations, and offload the management operations from the server's CPU and memory.

Add as many controllers as reasonable for your workload, connect your virtual disks accordingly, then spread out and best distribute your SQL Server object workload among the available disks and controllers.

These controllers have many of the components of a SAN-based controller, including processors and memory. As the connected storage becomes faster, many of these controllers suffer in performance. Controller CPUs can become overwhelmed. Cache levels are insufficient. Even the backplane connectivity could be insufficient. The internal I/O queues could also be too small. For this type of storage connectivity, ensure that your storage controller is the most robust you can afford.

VMs require storage controllers as well. These controllers also have their own individual performance qualities and challenges. The aforementioned VMware PVSCSI controller has a queue depth default of 64 and can be overridden up to 254. The LSI SAS controller, the default controller type for VMware® Windows-based VMs—and Hyper-V's only option—has a non-adjustable queue depth of 32.

VMware also has a newer option: the NVMe controller. It was introduced in vSphere 6.5, and can boost performance for NVMe-based storage types.

VMs also allow for the addition of multiple disk controllers per VM. You can add up to four SCSI controllers to VMware and Hyper-V VMs.

VMware also allows for adding up to four NVMe controllers in addition to the other types. SATA and IDE controllers can be leveraged as well, but the queue depths are much lower than the other types, and not recommended for SQL Server workloads.

Add as many controllers as reasonable for your workload, connect your virtual disks accordingly, then spread out and best distribute your SQL Server object workload among the available disks and controllers.

Simply moving your SQL Server databases to an all-flash SAN might not bring the performance improvement you want. To find out, first review the infrastructure, looking for additional bottlenecks preventing the all-flash SAN from kicking into high gear and flexing its muscles.

Network-attached storage, such as the Windows iSCSI initiator, doesn't contain the block-level queues; but that does not mean you're free of queues. The iSCSI StorPort driver has a default value of 20 for physical miniports and 250 for virtual miniports for the initial LUN queue depth.

Controller type selection for physical and virtual machines is quite important for the performance of the SQL Server databases.

Tuning for Flash

All-flash SANs provide extremely low latencies when compared to their spindle counterparts. This low latency will most likely help improve the SQL Server performance if the storage layer is a current bottleneck to performance (and most of the time, it is).

However, simply moving your SQL Server databases to an all-flash SAN might not bring the performance improvement you want. To find out, first review the infrastructure, looking for additional bottlenecks preventing the all-flash SAN from kicking into high gear and flexing its muscles.

Once the infrastructure bottlenecks are removed, it's time to dive into SQL Server storage internals.

Spreading out the workload could make an improvement to overall performance. If the storage layer is no longer the performance bottleneck, the bottleneck shifts farther up the infrastructure stack. It could even make it all the way into the Windows OS layer. You'll see a symptom of this challenge when the SAN performance latency metrics are quite low but the Windows layer Perfmon metrics for disk latency are quite high.

If the interconnect layer has been ruled out as a problem, the Windows I/O queues per disk controller could be the bottleneck. Adding additional disks with additional disk controllers, each with their respective queues, could relieve some of this pressure and reduce latency.

SQL Server can only make changes to data (insert, update, and delete operations) as fast as it can write a record of that change to the transaction log file. All-flash can make the latency of this operation very small, improving the performance of the change operation. This log stream is in a sequential write stream to disk.

Dedicating a disk controller and separate disk or LUN for database logs can help with the performance if the volume of writes is too high.

TempDB is the SQL Server container for all the random tasks that occur on a routine basis, such as sorting, temporary objects, spooling, and database integrity checks. Placing this database's data and log files on the all-flash array can also improve the performance of any operation, either read or write, that makes heavy use of TempDB.

Contrary to common belief, database index maintenance is still critical to keeping a healthy database. Proper maintenance keeps the database layer efficient. Moving the database indexes to a different file group and data file, and placing this file on high-performance media and LUN, can also improve performance.

Tuning for NVMe

Tuning your SQL Servers for NVMe-based storage continues the trends of flash. Whereas better distribution of the SQL Server workload across numerous database files, disk controllers, virtual disks, and paths to storage is sure to improve performance of the workload, this design pattern is almost a requirement for NVMe-based workloads.

The numerous queue counts and high commands per queue available on NVMe storage lends to better performance for a wider and more distributed workload.

To make the best use of the new NVMe standard, the wide distribution of SQL Server objects should be designed into the platform architecture from day one to best distribute the workload across all available queues between the database and the storage.

Workload Storage Handling Bottlenecks

Every IT platform has bottlenecks. The most pressing question on most administrators' minds is the impact of the bottleneck on a particular process or application.

That brings a related concern about where the next bottleneck will pop up when the current one is resolved, and worry over the likely severity of that one.

As the underlying storage bottlenecks are improved, the next bottleneck that organizations usually encounter with database performance

is how (in)efficiently the queries and database tables are architected. Some of the common issues include:

- Tables created with poor choices in data types
- Missing or very large primary keys
- Clustered index columns either missing or also very large
- Foreign keys are undefined.

Queries are usually constructed in such a way that development stops when the results produce the desired output. But rarely, if ever, is a review done of the underlying impact of the requirements to fulfill the request in the appropriate amount of time.

Many times, these inefficient database designs and access patterns result in a very high demand on the underlying platform. CPUs can be crushed with widely parallelized tasks that drain vital compute power. More likely, the database access requires orders of magnitude greater I/O demand to fulfill the query. And all because of inefficient design and access.

As the underlying bottlenecks in storage performance and queueing are improved, these design inefficiencies will result in a more apparent performance penalty when encountered.

The scalability of your product depends on the efficiency of the data layer. Efficiency in design and queries will dramatically improve the performance of the application and improve the experience for end users.

For applications that have full control over the code, focus on database design and query efficiency just as much as the focus for new features and functionality. The scalability of your product depends on the efficiency of the data layer. Efficiency in design and queries will

dramatically improve the performance of the application and improve the experience for end users.

Sometimes, however, business applications are sourced from third-party software vendors, and organizations can rarely modify the database design and/or queries that originate from the application.

In these circumstances (subject to the supportability of the ISV), minor efficiencies can be gained with tweaks. These can include advanced instance-level modifications, index and statistics optimization, and concurrency model adjustments.

Most organizations, though, choose to maintain performance by increasing the hardware performance underneath the application and databases.

For many storage-bound and time-sensitive database requirements, increasing the storage performance could be a cost-effective method of improving application performance.

SQL Server Storage Requirements and Budget

Most data volumes grow exponentially, on a yearly basis. Your organization is sure to be no different; businesses collect more data from more sources and want this data online for longer, to help gain fresh business insight.

When businesses plan for data and growth strategies, the common process is to project the amount of space needed a certain number of years in the future.

Capacity is only half of the challenge; storage performance is also key to maintaining that data. Trending the performance metrics next to the data growth rates will help you determine if your storage can deliver the necessary performance down the road.

For example, Figure 6-4 shows a sample projection from a business. This business has been rapidly adding data into their environment over the last five years.

Recently, they purchased a new SAN they expect to last for the next hardware cycle of five years. It contains 80 TB of usable space. In their projections, they will easily be able to store their data within this footprint through 2020, when the next purchase cycle is due.

So there's enough capacity, but a problem looms on the horizon: The SAN they purchased can sustain a maximum IOPS of 500,000.

Based on the historical data and projections, the storage will effectively run out of performance around the middle of 2021. Beyond that point, the performance of the array will suffer as data continues to be added, and the business will be negatively impacted by this imminent performance bottleneck.

When budgeting for the next storage purchase, take into consideration the performance necessary to maintain the business-critical processes that run on the storage. Purchase the fastest array you can afford that can meet these requirements.

To avoid this situation, measure the ongoing performance metrics from within the different infrastructure layers around the clock, and use these to project the storage performance requirements—not just the capacity requirements.

When budgeting for the next storage purchase, take into consideration the performance necessary to maintain the business-critical processes that run on the storage. Purchase the fastest array you can afford that can meet these requirements.

Up Next

Now that you know how to design and tune the storage layer for maximum performance, let's focus on keeping you current. Newer SQL Server versions can be used to improve the efficiency and performance of the storage layer. In the next chapter, you'll learn about some features of SQL Server that can be used in conjunction with your storage to create more benefit for the business.

CHAPTER 7

Modernizing SQL Server

If you haven't yet done it, today is the right time to start upgrading your SQL Servers to the most recent version, as well as improving the infrastructure underneath it to provide the foundation for peak performance from the applications.

Features in the Latest Version

SQL Server on Linux

SQL Server now runs on Linux! New (as of the SQL Server 2017 release) SQL Server on Linux allows for the deployment of enterprise, production-grade deployments on Red Hat Enterprise Linux, SUSE Linux Enterprise Server, and Ubuntu.

The vast majority of SQL Server on Windows features are fully supported, including availability features such as Availability Groups and FCIs. The performance characteristics of SQL Server on Linux are equal to the Windows OS equivalent.

Microsoft improved SQL OS, the miniature operating system inside the SQL Server engine. The update established the performance abstraction layer (PAL), which allowed the platform to be modified to run on Linux with (relatively) little effort.

The vast majority of SQL Server on Windows features are fully supported, including availability features such as Availability Groups and FCIs. The performance characteristics of SQL Server on Linux are equal to the Windows OS equivalent.

Customer choice has also been enhanced, since SQL Server on Linux allows IT organizations to have greater flexibility in the platform of choice for Microsoft's flagship database engine.

All the performance-oriented recommendations for SQL Server on Windows directly apply to SQL Server on Linux deployments, including storage queueing, object placement, and VM configuration.

SQL Server in Containers

Along with the native Linux support in SQL Server 2017, SQL Server now can run in a Docker container. Container support means SQL Server can now run on Linux, Apple, and even back on Windows platforms (including support for Kubernetes.)

Rapid development initiatives and methodologies such as DevOps can now rapidly provision and automate SQL Server deployments with ease.

Improved Scale-Out with Availability Groups

Each version of SQL Server has incorporated improvements in scale-out read workloads through the use of Availability Group (AG) read-only routing.

Secondary AG replicas can be configured to process read-only queries for active databases, such as reporting and analytics workloads, to better offload this workload from the primary AG replica.

The Availability Group listener can automatically route this type of traffic to secondaries to make the scale-out activity virtually transparent to the application servers. The entire workload scales outward,

and organizations can make better use of the SQL Server platform if the workload demands this level of scale.

In-Memory OLTP

SQL Server 2014 released In-Memory OLTP, a feature that allows the DBA to move individual tables to a new in-memory layer. The performance boost can be quite substantial (as much as 30x faster), but this feature is dependent on the storage layer. Over the last few releases, improvements to its performance and scalability have continued.

While the working set of data stays resident in memory, any changes must be written to disk before the operation completes. The faster the storage layer, the faster the operation completes, and the faster the application continues with its work.

SQL Server 2014 released In-Memory OLTP, a feature that allows the DBA to move individual tables to a new in-memory layer. The performance boost can be quite substantial (as much as 30x faster), but this feature is dependent on the storage layer. Over the last few releases, improvements to its performance and scalability have continued.

Furthermore, if a SQL Server needs to restart, all the data in these in-memory tables must be read from disk before the database is ready to use. Reading hundreds of gigabytes (or more) of data will take a lot of time on slower storage, possibly leading to an extended outage.

As a result, if your organization is considering adopting In-Memory OLTP for your applications, ensure that you have the fastest possible storage that fits within your budget.

Clustered Columnstore Indexes

Designed for OLTP tables with more than 10 million rows of data, columnstore indexes create indexes column-by-column instead of row-by-row, which allows developers to produce data warehouse-style analytical queries on OLTP workloads.

SQL Server 2016 and 2017 improves upon the columnstore index feature by adding the ability to have updateable nonclustered columnstore indexes and columnstore indexes on a memory-optimized table; additionally, columnstore indexes can now have a non-persisted computed column.

Benefit Analysis of Upgrading

Support

First and foremost, if you're currently running older versions of SQL Server, check to see if your version is covered by Extended Support.

- **SQL Server 2012.** Mainstream support for SQL Server 2012 ended in July 2017, and extended support expires on July 12, 2022. While this date feels safely in the distance, it will be here before you realize.
- **SQL Server 2008 and 2008R2.** Extended support for SQL Server 2008 and 2008R2 will end on July 9, 2019. In other words, it's almost here!
- **SQL Server 2005 and older.** If you're unfortunate enough to have a 2005 or older version of SQL Server in your environment, the extended support dates have passed; if that's the case, you should drop what you're doing and work to get these instances upgraded as soon as possible.

Expired support means that if you have any production outage, encounter bugs, or have general help, Microsoft could refuse to help you.

This isn't a position your organization should ever be in. The upgrade path is straightforward. You'll need to work with your business to demonstrate the value in the upgrade.

New Features

If your business needs additional reasons to upgrade legacy installations, just look at the new and updated features that come with the latest version of SQL Server.

- SQL Server on Linux and containers, for improved flexibility and automation
- "Always On" Availability Groups for improved high availability and disaster recovery
- In-Memory OLTP for greatly improved performance
- Improved Columnstore index features
- Improved query processing and optimization
- Backup encryption support
- Support for greater numbers of CPU and memory
- Improved security
- Improved T-SQL functionality

Widespread Business Benefits

The added power provided by the newest version of SQL Server is sure to improve the performance and availability of your database servers.

This means that productivity will increase. Data availability will improve. Administration overhead will decrease. Security will improve. Taken together, it's clear that the business benefits of the upgrade will be felt by all.

Plus, if your environment has the volume of SQL Server instances that tip the scales for license reductions through virtualization, you can receive all the features listed previously while simultaneously reducing the ongoing operational costs.

Maximizing SQL Server with Flash

All-flash SANs present a two-fold challenge. The upfront costs of the SAN can be daunting for some organizations, and the business case must be made to justify the purchase. Once acquired, the environment must be adjusted to make the most of this new storage layer.

Perception and Justification

One hurdle you'll likely need to overcome within your organization is the perceived separation of dependencies between SQL Server and storage.

To help show the benefit of all-flash storage to the DBAs and application owners, you need to turn an existing performance challenge with your existing storage into a business proposition.

Management generally goes cross-eyed if you start to describe challenges such as "thin provisioned storage," "fragmentation," "block sizes," or "fast" or "slow" without quantifying it all. There might be a feeling that you just want a shiny new toy to tinker with.

That's why it's imperative that you put the business first in the proposition. Here are some compelling arguments toward that end:

- "The business users are not satisfied with the speed of the application, and the storage is holding back the application."
- "Each user spends up to two full hours each day waiting on the system to execute a task, and 80% of these delays are because the system is waiting on storage."

- "Faster storage will improve user productivity by up to 1.5 hours for each employee, each day."

The justification you present should show the value of the storage to the business in terms of lost productivity.

Adoption

Once acquired, a flash storage array can be installed and your data migrated. At this point, the infrastructure bottlenecks will shift.

How they shift depends on your topology and environment, but they are guaranteed to shift. Network adapters can now reach peak throughput, causing a slowdown. HBAs can create latency at the server through queue bottlenecks, and so on.

At this point, you should be ready to review the tips presented in this book and be able to work to correct any internal bottlenecks. Your goal is to shift the bottlenecks back onto the SQL Server database and application layers, and not have the infrastructure hold either of these back.

That's a Wrap!

We hope you enjoyed this introduction to SQL Server and how it uses storage; its workload properties; and how to tune the storage to get the best performance for the database.

We hope your journey here has helped you to better understand the intersection of these two technologies and how one layer matters to the other.

Remember that this is just an introduction to SQL Server. There's so much more richness that you can mine for your operations. Your next step should be to go here to learn more:

https://www.westerndigital.com/solutions/application-workload#microsoft-sql-server