Storage Spaces - Designing for Performance

- Article
- 01/17/2024

Abstract

With Storage Spaces, Windows Server 2012 provides the storage performance, flexibility, and scale required by today's enterprise workloads. This paper focuses on the parameters that impact the performance of Storage Spaces and how these parameters can be optimized for specific workloads and enable Storage Spaces to achieve optimal performance.

Note: For the latest Storage Spaces design guidance, see Software-Defined Storage Design Considerations Guide.

This paper applies to the following operating systems:

- Windows Server 2012 R2
- Windows 8.1
- Windows Server 2012
- Windows 8

The suggested audience for this paper is:

- Storage Administrators
- Information Technology (IT) Professionals

Introduction

The performance of a storage solution is important when evaluating its cost efficiency and usefulness. Storage Spaces is a storage platform that was designed with performance as one of its primary goals. The premise was to deliver what the underlying hardware could provide. There exist a large number of variables that affect performance, responsiveness, and resiliency. Most notably, the characteristics of a given workload, such as Microsoft SQL Server, Microsoft Exchange, or virtual desktop infrastructure (VDI), determine how the storage can be optimally configured. In some scenarios raw throughput – often measured in MB/s – is the most important factor, while workloads with random data transfer patterns typically measure effectiveness in terms of input/output operations (I/Ops). Storage Spaces is capable of being a storage back-end to a variety of workloads. This paper describes how you can configure Storage Spaces for optimal performance and what hardware to select for a given workload.

For a glossary of common storage terms, see the end of the paper.

Storage Space Types

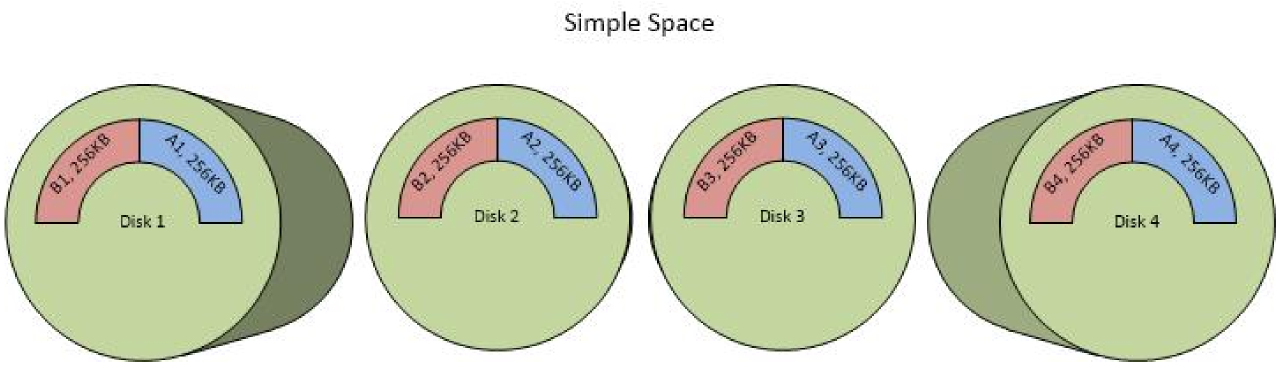
Windows Server 2012 introduces Storage Spaces, which is a new capability in Windows that enables the aggregation and virtualization of physical disks into logical groups, known as storage pools and storage spaces. You can choose the resiliency type of a storage space based on business needs, with choices of Simple (no resiliency), Mirror, and Parity. Below is a short explanation of how each resiliency type works.

For an introduction to Storage Spaces, please see this blog post: http://go.microsoft.com/fwlink/?LinkId=260762

Simple storage spaces

The concept of striping is most easily explained using the example of a simple space. Striping refers to the idea of writing data across multiple disks to reduce access and response times. To use a simplified example: imagine you have four disks and 1 MB of data to write to these disks. You could either write all of the data to a single disk and access it from there, or write 256KB to each of the four disks simultaneously, resulting in a 4x decrease in write times. This general principal holds true for sequential, as well as random workloads. The more disks the storage space can stripe across, the better the performance will be. Note however that this can only be realized if your applications have enough data outstanding. Striping in storage spaces can be manipulated by the *NumberOfColumns* and *Interleave* parameters – explained later in this paper.

The diagram below shows a simple example of striping. Using the example of a simple space, you can see that 256 KB are allocated on each of the four disks to write a total of 1 MB of data. A1 through A4 represent that data in the diagram. The next write to this storage space will allocate another four 256 KB chunks and write data to it, depicted by B1 through B4. Note that the 256 KB chunks are not necessarily in the same location in the below diagram:



Simple spaces are not resilient to disk failures and should thus be used only for temporary data and data that can easily be recreated, for example intermediary video rendering files or caches. From a performance standpoint, simple spaces provide the best overall performance when considering reads and writes. They provide the cumulative performance of multiple disks, depending on how many disks are in the storage pool.

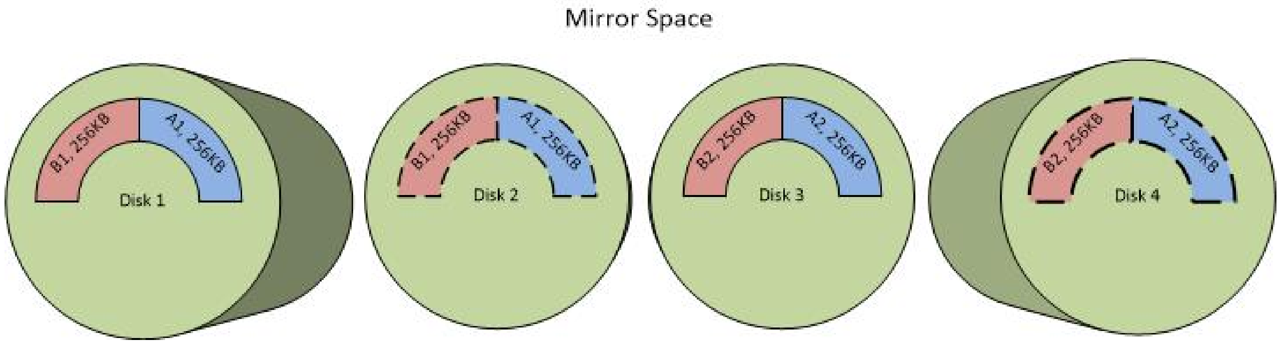
Mirror storage spaces

Mirroring refers to creating two or more copies of data and storing them in separate places, so that if one copy gets lost the other is still available. Mirror spaces use this concept to become resilient to one or two disk failures, depending on the configuration.

Take, for example, a two-column two-way mirror space. Mirror spaces add a layer of data copies below the stripe, which means that one column, two-way mirror space duplicates each individual column's data onto two disks.

Assume 512 KB of data are written to the storage space. For the first stripe of data in this example (A1), Storage Spaces writes 256 KB of data to the first column, which is written in duplicate to the first two disks. For the second stripe of data (A2), Storage Spaces writes 256 KB of data to the second column, which is written in duplicate to the next two disks. The column-to-disk correlation of a two-way mirror is 1:2; for a three-way mirror, the correlation is 1:3.

Reads on mirror spaces are very fast, since the mirror not only benefits from the stripe, but also from having 2 copies of data. The requested data can be read from either set of disks. If disks 1 and 3 are busy servicing another request, the needed data can be read from disks 2 and 4.

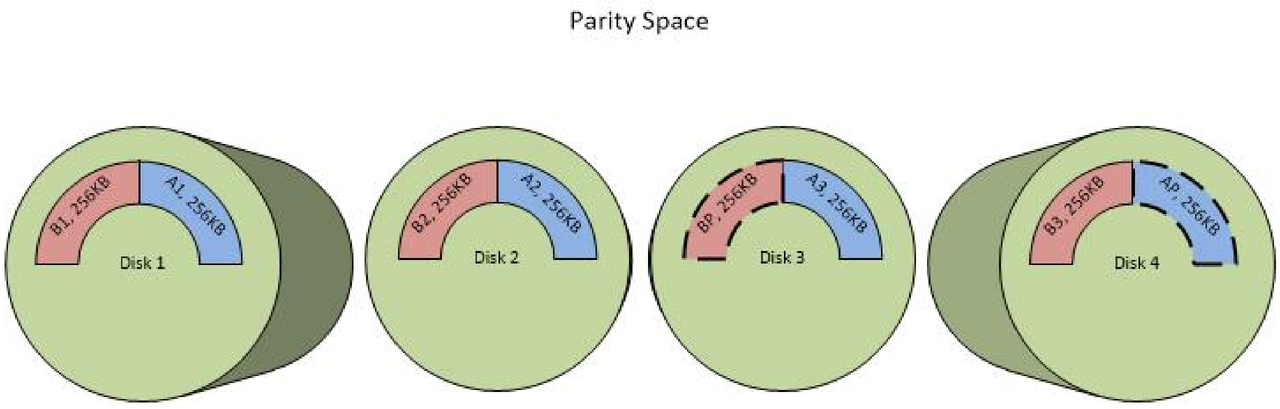


Mirrors, while being fast on reads and resilient to a single disk failure (in a two-way mirror), have to complete two write operations for every bit of data that is written. One write occurs for the original data and a second to the other side of the mirror (disk 2 and 4 in the above example). In other words, a two-way mirror requires 2 TB of physical storage for 1 TB of usable capacity, since two data copies are stored. In a three-way mirror, two copies of the original data are kept, thus making the storage space resilient to two disk failures, but only yielding one third of the total physical capacity as useable storage capacity. If a disk fails, the storage space remains online but with reduced or eliminated resiliency. If a new physical disk is added or a hot-spare is present, the mirror regenerates its resiliency.

Parity storage spaces

Parity leverages computation to recover data from a failed disk. One of its columns (explained later) in this type of storage space is used to store a parity-bit, which can be used to reconstruct data from a failed disk. Storage Spaces utilizes rotating parity which means that the parity bit for all stripes does not reside on a single disk, but that it rotates from stripe to stripe across different disks. In essence, every disk in a parity space contains data as well as parity information.

In the diagram below, a 768 KB stripe of data is written across disks 1 through 3 (A1, A2, A3), while the corresponding parity bit (AP) is placed on disk 4. For the second stripe of data, Storage Spaces writes the data on disks 1, 2 and 4, thereby rotating the parity to disk 3 (BP). Since parity is striping across all but one disk in the storage space it has good read performance while still providing resiliency to a single disk failure. Capacity utilization is more efficient than a mirror space.



The caveat of a parity space is low write performance compared to that of a simple or mirrored storage space, since existing data and parity information must be read and processed before a new write can occur. Parity spaces are an excellent choice for workloads that are almost exclusively read-based, highly sequential, and require resiliency, or workloads that write data in large sequential append blocks (such as bulk backups).

The sequential write performance of parity spaces can be improved by using dedicated journals. A journal is used to stage and coalesce writes to provide resiliency for in-flight I/Os. This journal resides on the same disks as the parity space unless you designate physical disks in the pool as journal disks. The journal is a mirror space and thus resilient by itself. The advantage of dedicated journal disks is a significant improvement in sequential write throughput for parity spaces. Incoming writes are de-staged to the parity space from dedicated disks, thus significantly reducing seeking on the disks used by the parity space. Using two SSDs as a dedicated journal to a three disk parity space, a sequential throughput increase of ~150% was observed under lab conditions. Note that the throughput of the journal disks will now be the overall throughput limit to all parity spaces created on this specific storage pool and you might trade extra capacity for performance. In other words, ensure that dedicated journal disks are very fast and scale the number of journal disks with the number of parity spaces on the pool.

Disk Types

The type of disk that is used for a given workload is important, as there are significant performance and capacity differences between them. The list below is a quick overview of what types of disks are generally available and what their characteristics are:

- **HDD – 2.5” disks** These “laptop” disks can run at a variety of RPMs, but generally are smaller in capacity and have less performance than 3.5” “desktop” or “server” hard disks. The tradeoff these disks make is in decreased power usage, and increased density in the number of physical disks per server.
- **HDD – 5,400 RPM and Variable RPM “Green” Drives** These disks run at either 5400 RPM or a variable RPM rate, and while they usually match 7,200 RPM disks in capacity, they provide reduced performance in exchange for decreased power usage.
- **HDD – 7,200 RPM** These disks usually provide the greatest single-disk capacity, and adequate performance, depending on the workload. Their sequential read and write performance is adequate, while random read and write performance is typically a fraction of SSD performance. Random workloads can quickly saturate the performance capabilities of these disks.
- **HDD – 10,000 RPM:** These disks balance high levels of performance with relatively large capacities, and somewhat higher prices. Their random read/write performance is significantly better than that of 7.2k disks while the sequential performance is slightly improved.
- **HDD – 15,000 RPM:** Continuing with the capacity / performance trend, 15,000 RPM disks are very quick at reading and writing data, even randomly. They achieve very good sequential throughput, though with reduced capacity than that of 10,000 RPM. These are oftentimes the choice for very I/O heavy workloads that do not require a lot of capacity, such as logs, or boot disks.

- **SSD:** Solid-state drives (SSDs) are in a class of their own that deliver extremely high random I/O performance, often a factor of 20+ over that of 15,000 RPM disks. Streaming large block throughput often does not exceed two times that of 15,000 RPM disks. If your workload involves random small block reads and writes, and your budget allows, SSDs are an excellent choice. There are different types of SSDs available in the market, with different designs that affect performance and longevity:
    - SLC-type: Typically capable of higher write speeds than MLC-type, and can typically sustain a higher amount of full disk re-writes and stress over the course of the lifetime of the disk. These disks typically have lower capacity and are more expensive than MLC-type disks.
    - MLC-type: This is the design of a typical SSD, with excellent read performance, good write performance, and capacity. MLC-type SSDs typically wear out more quickly than SLC-type SSDs under heavy write workloads. Once the disks become worn out their performance can decrease significantly.
    - “Enhanced”-MLC-type: Some vendors offer eMLC SSDs that strike a balance between SLC and MLC. This type of disk cannot sustain as many delete/program cycles as SLC memory, but significantly more than typical MLC. It is suitable for workloads that perform a significant number of writes, but not to the point where the disks are consistently writing out data. Usually these disks are priced higher than standard MLC and the implementation is vendor specific.

To ensure that SSDs do not wear out prematurely Windows, supports trim and unmap commands. The storage optimizer sends trim commands to SSD devices to maximize their longevity. Furthermore, if applications support trim and/or un-map, Windows honors those commands and passes them on to the underlying device.

Many storage systems contain mixed media types. This means that a few high capacity 7,200 RPM disks are available for storing backups, or data that is not accessed very often. 10,000 RPM or 15,000 RPM disks might be deployed to house databases or other data that needs to be accessed and manipulated relatively quickly. Lastly, SSDs could be found in such a system as well, storing data that has to be readily available or data that is very sensitive to access latency, such as transaction logs or parent VHDs in a VDI-type deployment. This type of provisioning is also possible with Storage Spaces and allows the user to take advantage of the various characteristics of different disk types without having to buy the most high-performing type of disk for the job. The system administrator can create pools of different disk types, or group different disk types in a single pool and manually select which disks should be used for which storage space. These types of heterogeneous storage deployments are fully supported by Storage Spaces.

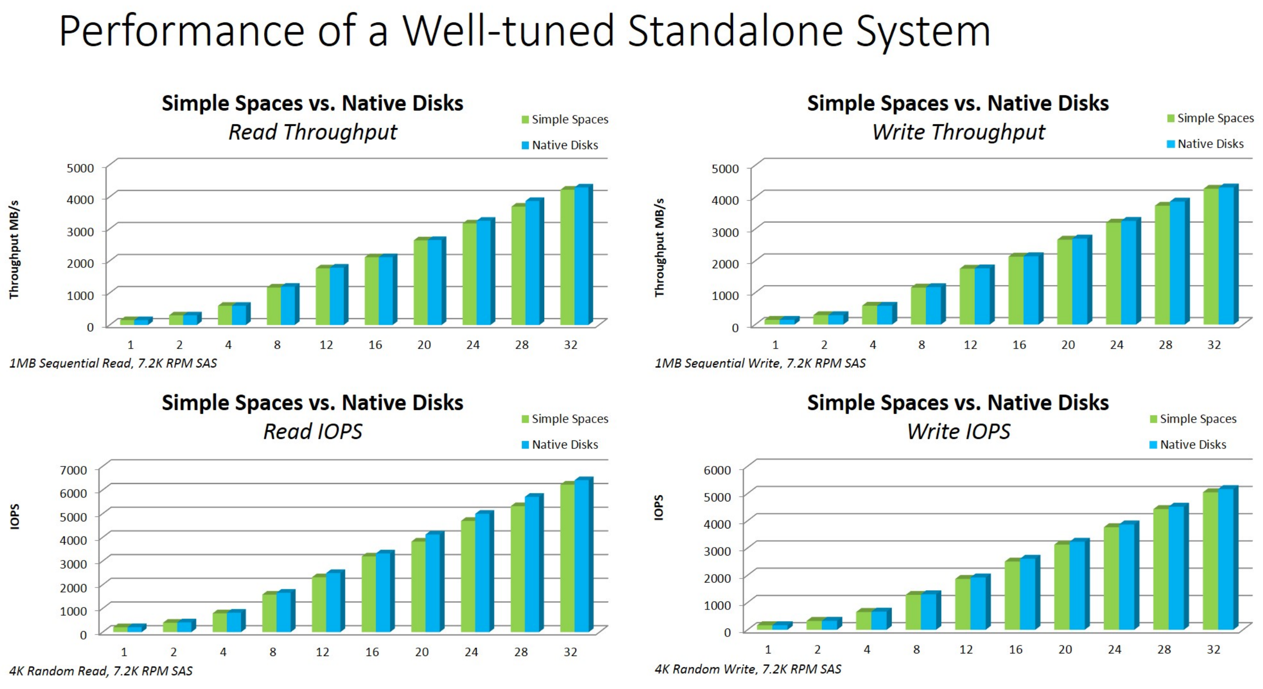
Supported Hardware Interfaces (SAS, SATA, USB)

Storage Spaces supports disks using SAS, SATA, and USB interfaces. SAS is typically considered enterprise-grade, while SATA equates to consumer storage. The major difference is that most SAS disks have two links to connect to, as opposed to just one for

SATA. This allows SAS disks to be shared resources and thus be used as storage in clustered and high-availability environments and be resilient to a single link failure. Additionally, SATA disks attached via a “just a bunch of disks” (JBOD) enclosure typically incur a slight performance penalty when compared to SAS. SATA SSDs further exacerbate the problem by having an order of magnitude larger CPU overhead. USB disks, while supported by Storage Spaces, typically do not qualify for high-performing system configurations. We recommend utilizing dual-port SAS disks when possible.

Scaling of Storage Spaces

Storage Spaces takes full advantage of the performance capabilities of the underlying storage. The following charts show the performance scaling of a Simple Space with up to 32 disks, with data striped across the disks in a simple space with no resiliency:



At TechEd North America 2012, as well as TechEd Europe 2012, Microsoft presented a three-node cluster system backed by 24 enterprise-level SSDs and Storage Spaces. The performance of this setup showcased a random read 1.4 million I/Ops and 10.9 GB/s of sequential throughput. More details about this system can be found in the Sample Deployments section of this paper.

Tiering (New in Windows Server 2012 R2)

Storage Tiering, the practice of moving frequently accessed data to very fast storage, while maintaining infrequently accessed data on moderate or slow storage is supported with Storage Spaces starting in Windows Server 2012 R2. Frequency of access (heat) on files is measured by the file system and fed into a tiering engine that instructs Spaces to move often used files to flash-based storage devices, while retaining cold data on large-capacity, slow storage. The major benefit is a significant increase in cost efficiency, as only the critical workload is accelerated by the flash-based storage, yet the majority of data stored can remain on slow, large-capacity devices (for example, 7,200 RPM 4TB HDDs).

To learn more about Storage Tiering, as well as sizing recommendations, please refer to the following TechEd content:

1. Best Practices for Deploying Tiered Storage Spaces in Windows Server 2012 R2 (http://channel9.msdn.com/Events/TechEd/NorthAmerica/2014/DCIM-B346#fbid=)
2. Delivering Exceptional IOPS/$ with Windows Server 2012 R2 (http://channel9.msdn.com/events/TechEd/NorthAmerica/2014/DCIM-B311#fbid=)

Block Size and Interleave

Two parameters that are important in any workload are block size and interleave (stripe size). Block size is a measurement (usually in KB) of the I/Os that the storage sub-system has to handle. Many database-style applications such as Microsoft SQL Server and Microsoft Exchange use relatively small blocks (8KB – 32KB), though other applications may use larger blocks. Sequential workloads profit significantly from larger block sizes. In general, IOPs and throughput are inversely related to block size: a small block size yields high IOPs with low throughput, while a large block size results in high throughput with low IOPS.

Storage Spaces can utilize multiple physical disks by striping data across them. One of the parameters that can be adjusted when creating a storage space is how large this stripe is supposed to be. The default setting is 256 KB. This means that Storage Spaces stores 256 KB of data on each column that was assigned to the storage space. If, for example, you have eight disks in the storage pool and want to write 4 MB of data, each disk receives two 256KB stripes (4MB / 256KB (Stripe) / 8 Columns = 2x 256KB stripes per column).

To maximize performance, ensure that the interleave value used by the storage space is at least as large as the I/Os of your workload. I/Os that exceed the interleave size are split into multiple stripes, turning one write into multiple writes, reducing performance.

Columns & Outstanding I/Os

The number of columns specifies how many physical disk data is striped across and has an impact on performance regardless of stripe and block size. Storage Spaces intelligently scales the column count up to eight by default, but you can adjust this parameter by using Windows PowerShell.

One example of how the number of columns affects performance is as follows. Assume that you create a simple (no resiliency) storage space in a storage pool with 12 disks, and that each disk is capable of 150 MB/s sequential throughput. Creating a simple space with one column yields a maximum throughput of 150 MB/s (the concept of spanning creates exceptions – see next section). Using two columns increases this to almost 300 MB/s, with further increases scaling linearly (for example, four columns yields close to 600 MB/s throughput). The more columns a storage space is assigned, the more disks can be striped across and the higher performance.

If you choose a high column count storage space and you intend to expand the capacity of the storage space in the future, you should add at least as many disks as the storage space has columns (in case of a simple or parity space), or columns times number of data copies (in the case of a mirrored space). For instance, when deploying a two-way mirrored space with four columns, you should have eight disks with available capacity (four columns times two data copies), and add disks in groups of eight when expanding the storage pool.

The following table provides real-world performance numbers comparing two simple spaces with different numbers of columns. In both conditions, the storage pool contained four SSDs, and each storage space received a sequential read request of 1 MB I/Os, akin to reading a large number of 1 MB files.

Disks	Total Capacity (TB)	Columns	Throughput (MB/s)	Avg. Latency (ms)
4	1.45	2	902.45	8
4	1.45	4	1622.50	4

The data shows that increasing the number of columns can significantly improve performance for sequential workloads, though at the expense of some flexibility when expanding capacity (you should add disks in groupings that match the number of columns x data copies). Random workloads do not experience as significant a performance increase, exhibiting more uniform performance across different column counts.

Another factor that ties in directly with the column count of the space is the amount of outstanding I/Os or data that is to be read from or written to the storage space. A large number of columns benefits applications that generate enough load to saturated multiple disks, but introduce unnecessary limitation on capacity expansion for less demanding applications.

The following example describes this trade-off: A typical hard disk is saturated writing a large number of 1MB files or block data in a sequential fashion. To saturate a simple space with four hard disks and the same workload, it is necessary to provide roughly 32 outstanding I/Os of this type. If less data is outstanding, not all disks are fully utilized. If the application using this storage space will never have more than 16 I/Os outstanding at a time, creating a storage space with two manually assigned disks could service this workload and preserve the other two disks for capacity expansion or for use with another application (using its own manually allocated storage space). This also has the advantage that the administrator can add two disks to extend the storage space as opposed to four.

To adjust the number of columns used by storage spaces, you must use Windows PowerShell. For more information, see: http://go.microsoft.com/fwlink/p/?LinkId=260763

Extended Performance Considerations

There are a few more points to consider and adjust when maximizing the performance of Storage Spaces. To get the most out of a storage subsystem it is important to ensure that the physical disks are the bottleneck and not the rest of the infrastructure, such as cables, HBAs, PCI slots, CPUs, or MPIO, as discussed below:

- **Cables and the SAS protocol:** SFF-8088 SAS cables are designed to transport up to 2.4 GB/s of data with 4-lanes of 6Gbps SAS (SAS Protocol 2.0). In practice, they typically reach their peak at ~2.2 GB/s. Hence, if a configuration is used that has many or very fast disks, it is important to deploy multiple cables not only for redundancy, but also for additional throughput. For example: 7,200 rpm disks servicing a sequential workload can reach roughly 150 MB/s per disk. Filling a JBOD with 24 bays up and running this workload means a potential throughput of 3.6 GB/s. Reaching this number will require at least 2 cables.
- **HBAs:** Host Bus Adapters (HBAs) connect the JBOD via SAS cables to a computer. Most HBAs are capable of moving 3+ GB/s of data and thus outperform a single cable. At the same time, most HBAs have at least 2 ports, meaning that 2 cables are theoretically capable exceeding the capabilities of a single HBA on sequential workloads. For high performance systems it is typically best to have one cable per HBA, two if MPIO is used for path redundancy.
- **CPU:** As long as hard disks are deployed CPU utilization is rarely a concern. If SSDs are deployed, it is possible for the CPUs of the server node to become the performance bottleneck. 24 high-performance SSDs can read randomly at over 1 million I/Ops., consuming a large amount of CPU resource. Two options are available to eliminate CPU bottlenecks from such configurations:
  - Use a server with more powerful CPUs or a greater number of CPU cores
  - Use multiple nodes in a cluster
- **MPIO:** Multipath I/O (MPIO) is a Windows Server 2012 capability that allows for multiple cables (paths) to be connected to the same set of SAS disks, providing path redundancy and load balancing. Using MPIO, it is possible to provide redundancy in the event of a failure of a cable, HBA, or JBOD SAS module. At the same time, load balancing can provide optimized performance by utilizing the capabilities of multiple SAS cables.

- **Power Protected Mode:** Storage Spaces allows administrators to enable “power protected mode”, which assumes that disk caches have batteries or are otherwise resilient to power failures. This mode can improve the performance of random workloads because metadata flushes to the disk are disabled and the disk has control over flushing of its cache. Note that enabling power protected mode without battery-backed caches or resilience to power failure puts your data at risk of loss or corruption.

Best Practices

As with all systems, there are a few important considerations for every deployment. Here is a short list of Storage Spaces best practices.

- Set your interleave to be at least as large as the most common I/O size from the applications that will be reading from and writing to the storage space. If you are unsure, use the default interleave size of 256 KB.
- Unless your workload has very specific needs and is unlikely to grow significantly, utilize the default column count chose by Spaces at creation time.
- When mixing disk types in the same storage pool, utilize manual disk selection (-PhysicalDisksToUse parameter) when creating a virtual disk, or separate different disk types into separate storage pools. Alternatively, utilize Storage Tiering (Windows Server 2012 R2)
- Do not use simple spaces unless resiliency is provided by the application or is unnecessary.
- Do not use parity spaces for workloads that are predominantly random in nature. Parity spaces are optimized for highly sequential / append-style workloads, such as archiving.
- When using dedicated journal disks for parity spaces, deploy SSDs.

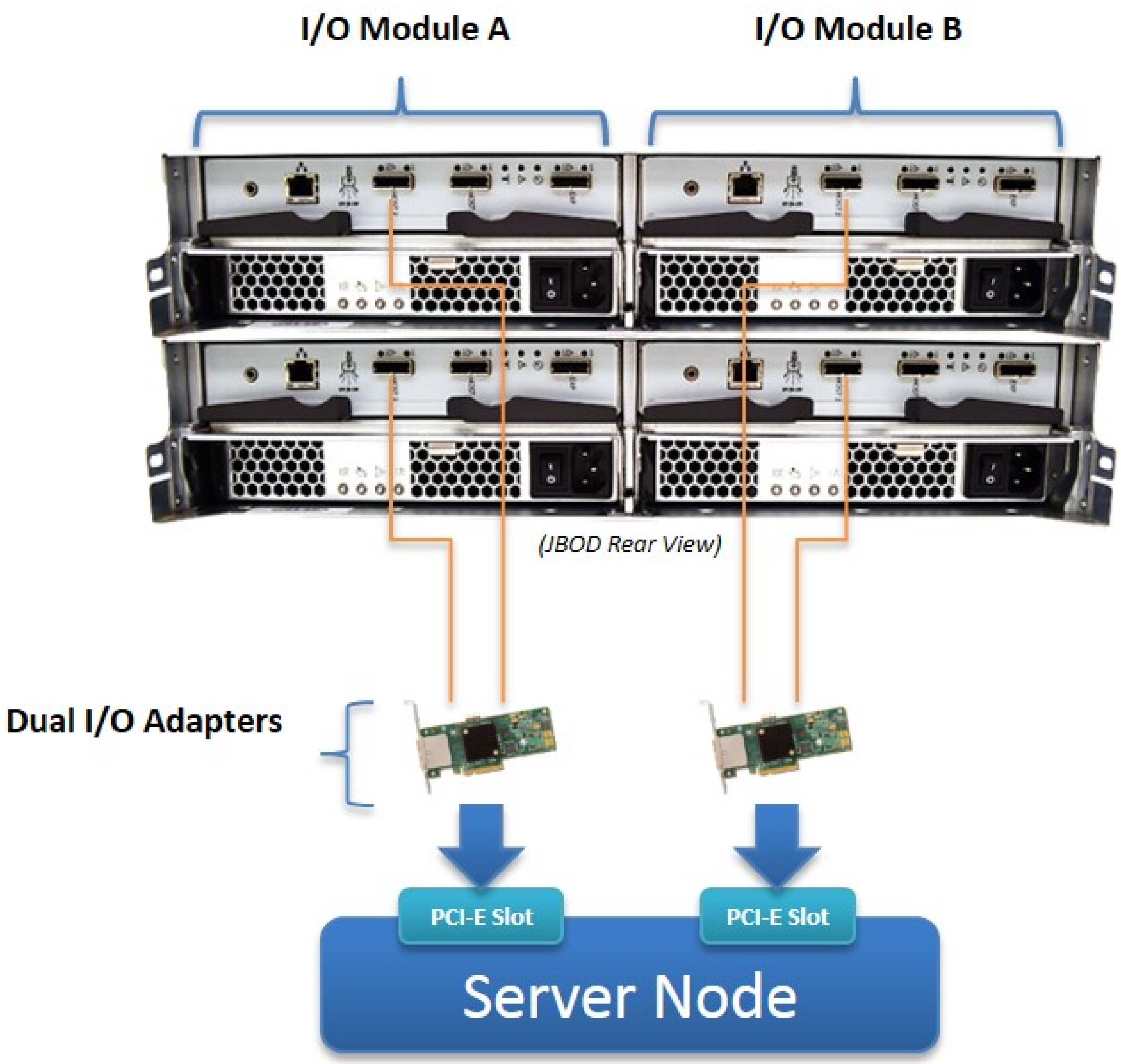
Sample Deployments

Standalone SQL Server Configuration

The following example describes how to deploy Storage Spaces in a standalone Microsoft SQL Server instance.

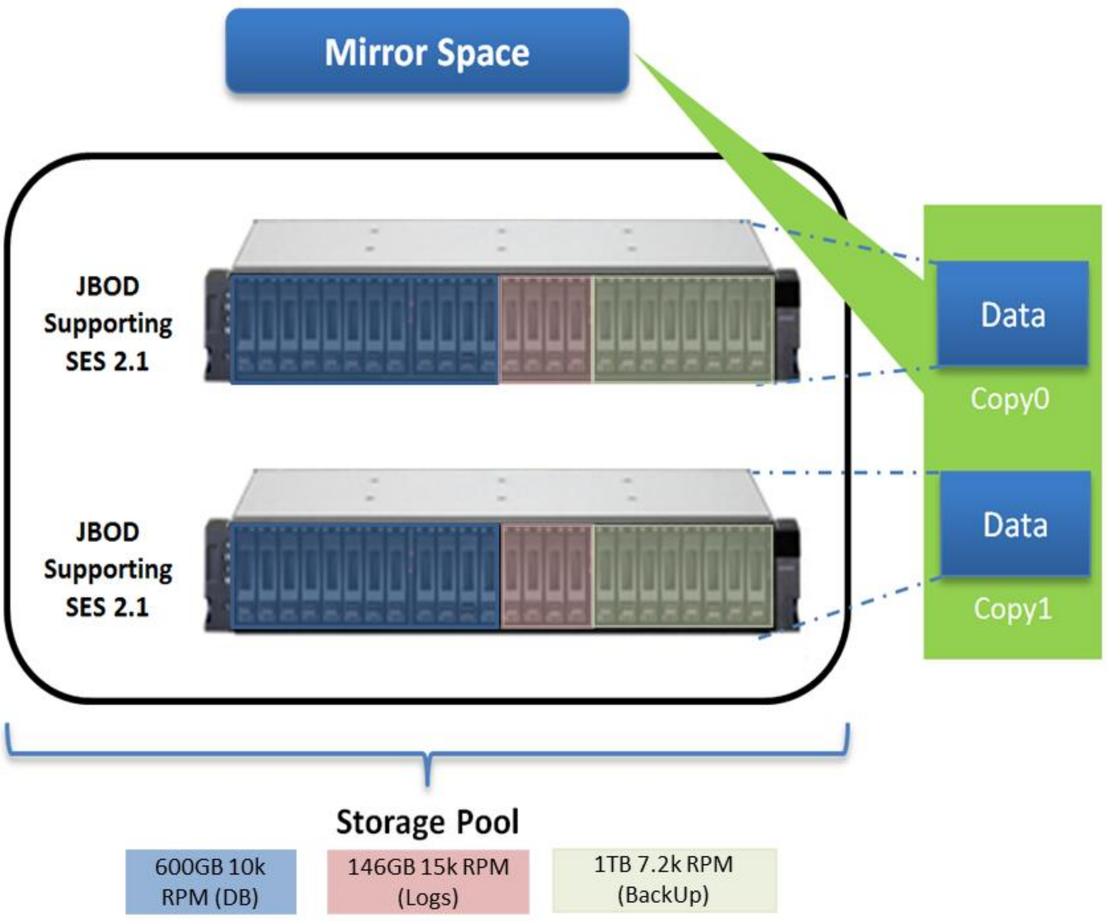
Server and Storage Connectivity

To build a resilient system it is important to have redundant paths from the server to the storage. The below diagram depicts how a standalone server can be connected to two JBODs in such a fashion that the system would be resilient to path-, HBA-, Disk-, and I/O-Module failure. If resilience to JBOD failure is desired, at a minimum, three JBODs have to be deployed and the "IsEnclosureAware" switch has to be used in PowerShell when the Space is created.



Storage Spaces Setup

The typical workload of an actively used SQL database is random in nature with relatively small block sizes (8KB – 32KB). The read / write percentage can vary significantly, based on what kind of application(s) the database supports. As a result of these characteristics, it is advisable to utilize fast disks (10,000 RPM or faster) and keep the storage space stripe size relatively small to accommodate the blocks read and written by the database. The diagram below depicts the **logical** view of how enclosure awareness functions. Two copies of the same data will never be placed on the same JBOD.



All 48 disks can be aggregated into a single storage pool. The active database tables and indexes can be stored on two mirror spaces with 12 10,000 RPM manually assigned and the following parameters:

Database Tables – Storage Provisioning (2 Mirror Spaces)	
Number of Columns	6
Number of Data Copies	2
Resiliency Setting	Mirror
Provisioning Type	Fixed
Interleave	64KB
Physical Disks To Use	<6 10,000 RPM disks from each JBOD>

This will allow for fast access times to the data and 7.2 TB of total capacity for the database tables.

Logs are important in SQL deployments and require relatively low (fast) read/write latencies to ensure smooth operation of the database. The SQL logs will be placed on a mirror space in a pool with 15,000 RPM disks from each JBOD. The parameters are almost the same as above:

Database Logs – Storage Provisioning (1 Mirror Space)	
Number of Columns	4
Number of Data Copies	2
Resiliency Setting	Mirror
Provisioning Type	Fixed
Interleave	64KB
Physical Disks To Use	<4 15,000 RPM disks from each JBOD>

Lastly, we need Space for backups of the database. Backup disks typically do not have to be very fast as the access pattern changes here. Backups are often in larger chunks and more sequential than day-to-day operations. Capacity is of importance though. The 1 TB, 7,200 RPM disks are a good capacity option and can be configured with the following parameters:

Database Backup – Storage Provisioning (1 Mirror Space)	
Number of Columns	8
Number of Data Copies	2
Resiliency Setting	Mirror
Provisioning Type	Fixed
Interleave	256 KB

Physical Disks To Use	<8 7.2k RPM disks from each JBOD>

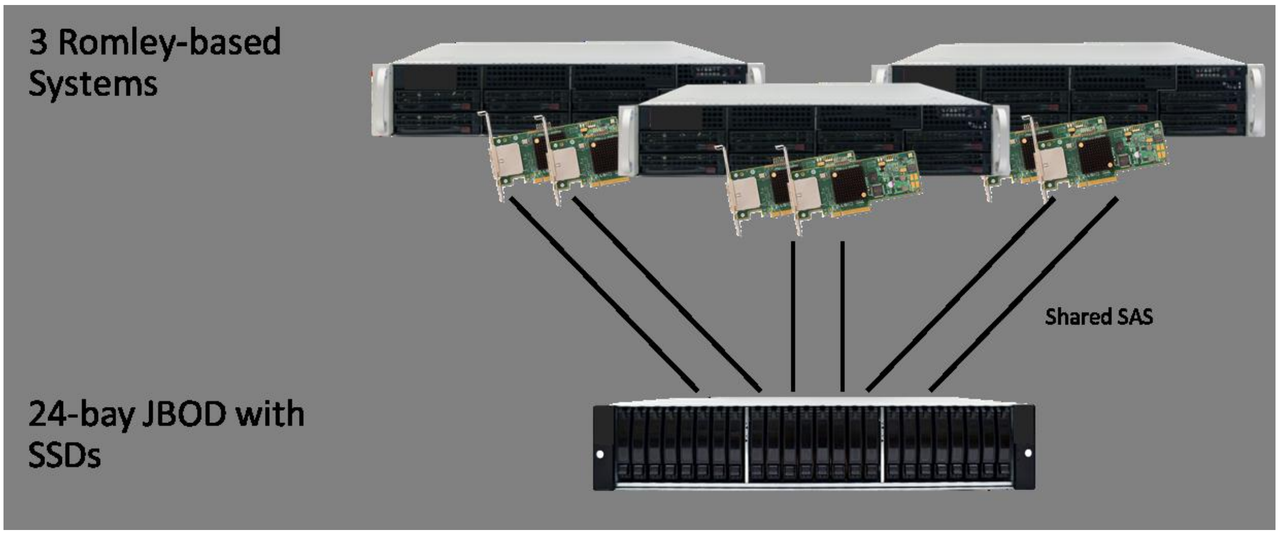
This provisioning scheme yields a total database capacity of ~7.2 TB, log capacity of ~584 GB, and ~8 TB of backup capacity while being highly focused on resiliency to component failure. Individual disk failures in each storage space can be tolerated, as well as adapter, cable, or JBOD failure. If this degree of resiliency is not required, then the provisioning can change to provide more performance or capacity.

Performance Cluster Element

This cluster deployment element is designed to provide extremely high levels of performance at a low price point. Over 1.4 million random read I/Ops and 10.9GB/s of sequential throughput have been demonstrated on the below hardware with Storage Spaces.

The below link is a recording of the TechEd talk presenting Storage Spaces in depth. The performance demo can be found in the talk “Windows Server 2012 Storage Solutions: Storage Capabilities for Everyone” at time 23:51: http://video.ch9.ms/teched/2012/na/WSV327.wmv

Server and Storage Connectivity:



Conclusion

As shown in this article, Storage Spaces is capable of scaling to multi-million I/Ops while satisfying many different workload types that will shape the behavior and configuration of the storage solution. Further, examples were provided of a near doubling in sequential performance when foregoing some flexibility. Storage Spaces has parameters that can be fine-tuned toward workload needs in order to achieve optimal performance for a wide range of different applications, thus allowing customization toward specific business needs. The use of commodity hardware makes Storage Spaces deployments flexible, easily expandable, and cost-efficient.

Microsoft has since released a Word document that outlines the various components and settings used to achieve the above 1M IOPS. It is well worth the read especially when it comes to the Iometer settings.

Microsoft Download Center: Achieving Over 1-Million IOPS from Hyper-V VMs in a Scale-Out File Server Cluster Using Windows Server 2012 R2

Glossary:

Below is a list of terminology used in this paper:

Term:	Definition:
Column	Columns correlate to underlying physical disks across which one stripe of data for a storage space is written. Typically the column count will be equal to the number of physical disks of the storage space (for simple spaces) or half of the number of disks (for mirror spaces). The column count can be lower than the number of physical disks but never higher.
Interleave	Interleave represents the amount of data written to a single column per stripe.
Block Size	Block size is a measure of how large or small, in kilobytes or megabytes a read or write request is. This is one of the most important parameters of a workload.
Outstanding I/Os	Also referred to as “Queue Depth”, outstanding I/Os measures how many read or write requests are queued up to be serviced by a disk, Space or other device.

Other Languages

存储空间 - 性能设计 (zn-CN)