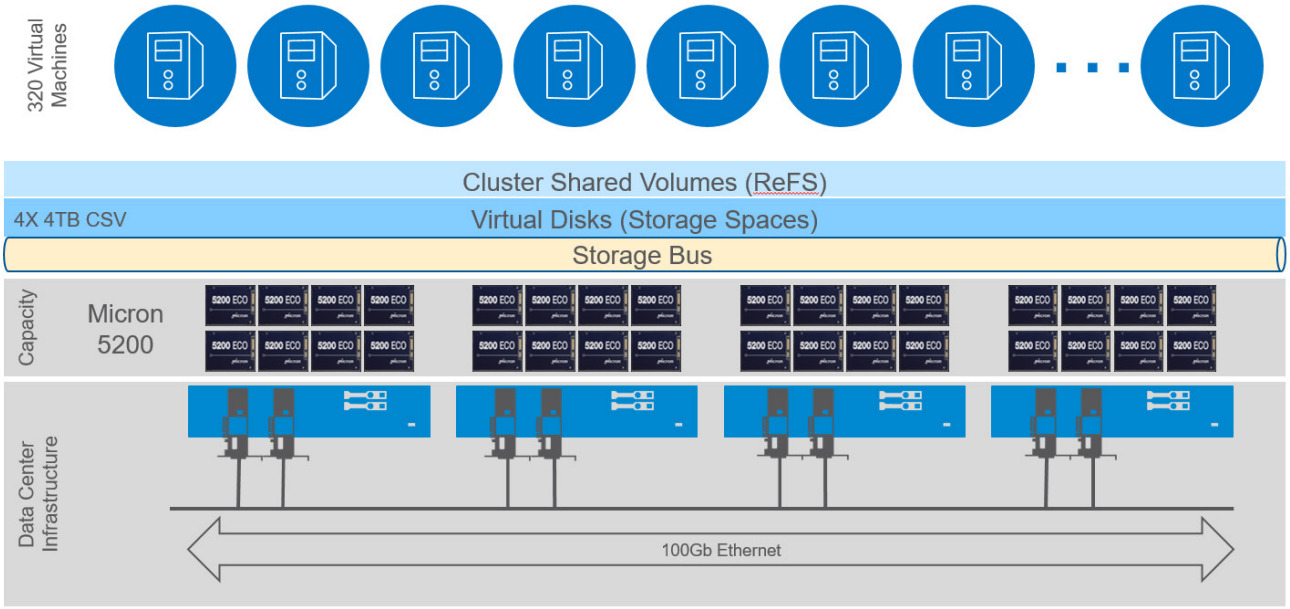
Hyper Converged Infrastructure Using Microsoft® Hyper-V® and Storage Space Direct with Micron® Enterprise SATA SSDs

Reference Architecture



Contents

Figures

Executive Summary

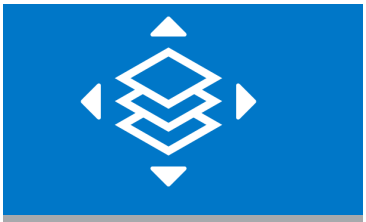
This reference architecture (RA) focuses on a hyper-converged infrastructure (HCI) solution using Microsoft Windows® Server 2016 with Hyper-V® and Storage Spaces Direct. It provides a blueprint for creating a four-node HCI cluster that can be expanded as needed to meet the needs of your applications. Included is a detailed performance analysis of the cluster for small- and large-block reads and writes for a fixed set of 80 virtual machines (VMs) per cluster node.

This RA illustrates a simple, single-tiered all-flash Storage Spaces Direct configuration using Micron Enterprise SATA SSDs and advanced DRAM to support Hyper-V VMs at scale. The performance test results show this solution can provide high performance at low latencies across a wide range of storage I/O profiles. The table below shows that using a standard 4 KiB block size with a total of 320 VMs running on a four-node cluster resulted in performance ranging from over 1.5 million IOPS for 100% random reads to more than 197,000 IOPS for 100% random writes. Across all small-block test scenarios, queue depth (QD) 4 per guest VM provided optimal performance with acceptable latency.

This document provides everything you need to build out a scalable high-performance Microsoft virtualized solution that is a great fit for Hyper-V, Azure®, and other Storage Spaces environments.

Read %	QD 1	QD 2	QD 4	QD 8
100%	1,010,827	1,339,248	1,521,334	1,561,441
90%	743,031	893,399	971,560	990,902
70%	462,559	501,168	503,299	490,139
0%	197,965	197,789	195,262	189,533

Table 1: IOPS by Guest VM Queue Depth



Micron Reference Architectures

Micron Reference Architectures are optimized, pre-engineered, enterprise-leading platforms developed by Micron with industry leading hardware and software companies.

Designed and tested at Micron's Storage Solutions Center by our software and platform partners, these best-in-class solutions enable end users, channel participants, independent software vendors (ISVs), and OEMs to have a broader choice in deploying next-generation solutions with reduced time investment and risk.

The Purpose of this Document

This document describes an RA for deploying a performance-optimized HCI using Microsoft's Hyper-V and Storage Spaces Direct software-defined storage (SDS) solution with Micron's Enterprise SATA SSDs. The hardware and software building blocks used to characterize performance are detailed in this document, covering the Micron HCI composition including the server and operating system (OS) configuration for a 4-node Hyper-V cluster. The purpose of this document is to provide a pragmatic blueprint for administrators, solution architects and IT planners who need to build and tailor high-performance HCI infrastructures at scale[1].

Why Micron for this Solution

HCI solutions like Microsoft's Hyper-V with Storage Spaces Direct are highly dependent on advanced, high-performance memory and storage to efficiently and effectively provide virtual server services to the enterprise. These important components — both SSDs and DRAM — can represent up to 70% of the overall component value of these solutions. Choosing the right components can determine your solution's ultimate success.

Micron's silicon-to-systems approach provides unique value in our RAs, ensuring these core elements are engineered to perform in highly demanding solutions such as HCI solutions like Hyper-V with Storage Spaces Direct and are holistically balanced at the platform level. By collaborating with customers on total data solutions, Micron develops and manufactures the storage and memory products that go into the enterprise solutions we architect.

1. Micron assumes no liability for lost, stolen or corrupted data or performance differences arising from the use of any Micron product. Products are warranted only to meet Micron's production data sheet specifications. Products, programs and specifications are subject to change without notice.

Solution Overview

Microsoft's HCI solution consists of two major components of their Windows Server 2016 operating system: Hyper-V and Storage Spaces Direct. Hyper-V is a virtual server solution supporting the ability to run multiple software-only servers on a single bare-metal server. Storage Spaces Direct is a software-defined storage solution that provides either traditional storage area network services for traditional data center solutions or a virtual storage area network that provides services to a HCI solution in conjunction with Hyper-V.

The Microsoft-based HCI solution described here consists of an example of a clustered deployment of four x86 industry standard servers. Each server consists of an all-SATA SSD virtualized storage solution that supports the VMs running within the cohosted Hyper-V server. Each node is interconnected using 100Gb Ethernet networking that provides communication services for all VMs, as well as advanced storage management within the Storage Spaces Direct implementation (Figure 1).

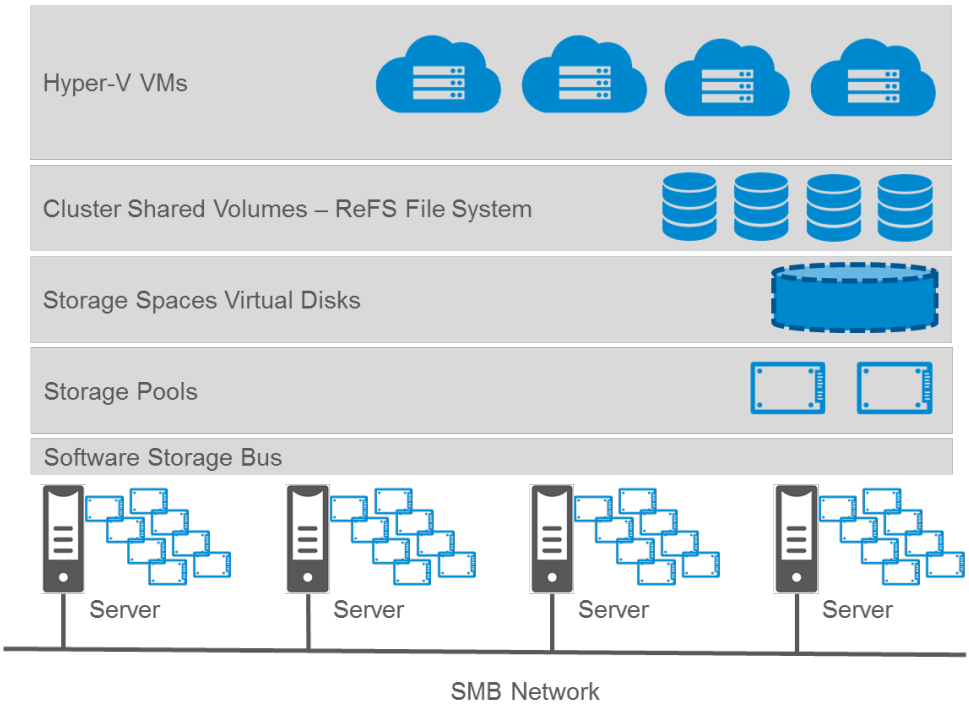


Figure 1: Microsoft Hyper-V with Storage Spaces Direct Solution Overview

Microsoft's Storage Spaces Direct solution consists of multiple layered components that provide a distributed storage infrastructure to provide scale-out, highly available storage services. Each server has one or more storage devices that are aggregated and managed by Storage Spaces Direct. While these storage devices can be spinning media or SSDs, this RA is designed to provide an all-SSD solution to maximize performance in a dense environment.

At the lowest layer is a distributed software-based storage service (Software Storage Bus). The role of the Software Storage Bus is to ensure that data is protected in the event of a Storage Spaces Direct target node failure. It does so by using a replication service that will store copies of all data on two or more Storage Spaces Direct nodes based on your resiliency requirements.

Above this storage bus layer, Storage Spaces Direct then allows you to organize the available storage in a flexible series of storage pools. Each storage pool can then have standard Storage Spaces virtual disks created to fit your application needs.

A key component of the high-availability of data is the use of the Resilient File System (ReFS). ReFS has several capabilities that make it suitable for an advanced hyper-converged SDS solution:

- ReFS uses checksums to detect data corruption. These checksums can be used for metadata only or for both metadata and file data.
- ReFS can automatically repair detected corruptions using an alternate copy of the data provided by the Storage Spaces services
- ReFS can scan each volume and identify latent corruption potential and proactively trigger repair of that data.

These features make ReFS a very robust file system for hosting data and large-scale VM environments.

Once the robust file system is in place, each ReFS volume can host application data or Hyper-V VM images. For this RA, Micron built a HCI solution for scale-out Windows data centers (Figure 2).

Based on this robust software architecture and our extensive experience in designing, developing and deploying solid state storage in enterprise solutions, as well as ongoing interaction with some of the largest enterprise environments and cloud providers, Micron selected Microsoft's Hyper-V with Storage Spaces Direct for this HCI SDS RA implementation. Microsoft's extensive market share in the enterprise data center as well as the simplicity, flexibility and manageability of its integrated HCI offering make it an excellent option for customers using Windows and Linux application servers.

Design Overview

The Micron HCI RA is composed of several primary components including servers, operating systems, software services and network infrastructure. These elements combine to create a robust, scalable HCI infrastructure for your Windows application services.

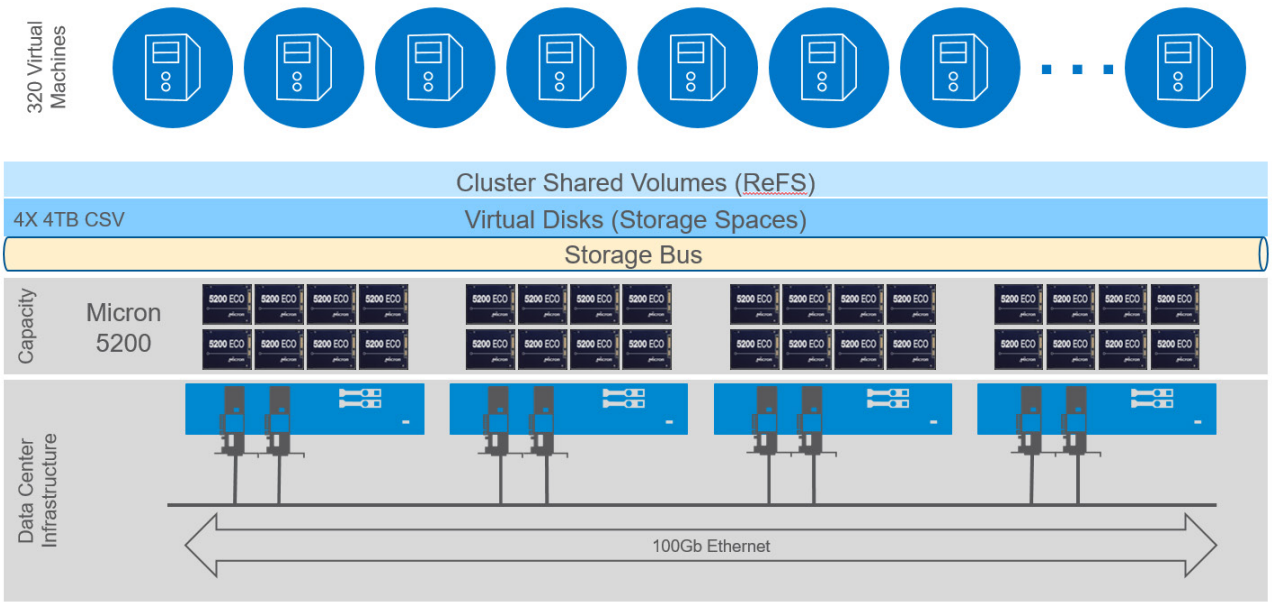


Figure 2: Micron's Hyper-Converged Infrastructure RA with Hyper-V and Storage Spaces Direct

Reference Architecture Elements

Each node within the RA is composed of several primary components:

- Industry standard x86 server based on Intel's Purley CPU architecture
- 384GB Micron DDR4 ECC RDIMM Memory
- Microsoft Windows Server 2016 Datacenter
  - Microsoft Hyper-V Server
  - Microsoft Storage Spaces Direct
- 1.92TB Micron Enterprise SATA SSDs (5x00 family)
- 100Gb Ethernet Networking infrastructure

Microsoft Hyper-V and Storage Spaces Direct Server Nodes

The RA consists of four nodes configured with Microsoft Windows Server 2016. Each node is configured with Microsoft's Hyper-V and Storage Spaces Direct services. Hyper-V is designed to provide scalable virtual server services that enable you to more efficiently utilize the full capabilities of your server resources.

In support of the Hyper-V service, each node is also configured with Microsoft's Storage Spaces Direct virtual storage service. Storage Spaces Direct can be configured as a standalone software-defined storage array supporting external physical and virtual servers, or it can be deployed integrated with Hyper-V to provide virtual storage service as part of a hyper-converged infrastructure solution. This RA uses the latter case.

High-bandwidth Ethernet Storage Network

Storage Spaces Direct is built upon a high-performance, remote direct memory access (RDMA) networking infrastructure for all data movement and data consistency functions. For this RA, advanced RDMA over Converged Ethernet (RoCE) functionality is used for all Storage Spaces Direct data management. The RoCE functionality is provided by a 100Gb Ethernet network switches and server network interface cards (NICs) from Mellanox®.

Micron Enterprise SATA SSDs

Micron's family of enterprise SATA SSDs provides workload-focused performance, endurance and capacities for read-centric, write-centric and mixed-use applications. Features include:

- **High Capacity:** Up to 7.68TB of storage in a 2.5-inch form factor and 1.92TB in an M.2 form factor.
- **Consistent High Performance:** Three endurance models are optimized for varying workloads, offering consistent, steady state random writes up to 74,000 IOPS (exact IOPS will depend on drive model, capacity and form factor).
- **Enabled Security:** Built-in AES-256-bit encryption and TCG Enterprise protection.
- **Ultimate Flexibility:** Micron's FlexPro™ firmware architecture enables you to actively tune capacity to optimize drive performance and endurance.
- **Outstanding Reliability:** 99.999% quality of service (QoS)[1] reduces downtime and latency.

Micron 32GB DDR4 ECC RDIMM Memory

Micron SDRAM modules provides the performance and reliability required for a wide range of mission-critical applications. For this RA, each Windows Hyper-V server node was configured with 384GB of RAM using 32GB DDR4 RDIMMs.

[1] Micron 5100 MAX 960GB SSD was measured against a competing SAS 12 Gb/s 15K 600GB HDDs. Performed on Intel* Core i7-4790K @ 4.0GHz, Asus* Maximum VII GENE motherboard, CentOS* 6.5 64-bit, FIO* 2.2.6, Workload - 4KB Block Size, 100% read, 100% random, queue depth 32, 99.999% Quality of Service. *Other names and brands may be claimed as property of others.

Solution Design — Hardware

Hyper-Converged Storage Spaces Direct Node (x4)

The RA is built upon an expandable base configuration of four x86 servers platforms; each server having the following configuration:

Server Configuration	
Form Factor	2 rack units (2RU)
CPU	Model: 2x Intel® Xeon® Gold 6148 processor Cores: 40 (20 physical cores per socket) Hyperthreading (2 virtual cores per physical core)
Server BIOS	U30 v1.36
RAM	384GB DDR4 (12x32GB RDIMM @ 2666 MHz)
PCI Express Used	1x8: HPE Smart Array P816i-a SR Gen10 (pass-through mode) 1x16: HPE IB EDR/Ethernet 100Gb 2-port (840QSFP28) 1x16: HPE IB EDR/Ethernet 100Gb 2-port (840QSFP28)
Power	Redundant
Operating System	Windows Server 2016 Datacenter Core (10.0.14393)
Storage	
Expander	12Gb SAS Expander
Boot Drive	240GB M.2 Micron 5100 PRO SSD
Data Drives	8x 1920GB Micron 5200 ECO SSDs
Network	
Network Interface Controllers	2x HPE IB EDR/Ethernet 100Gb 2-port (840QSFP28) (PCIe x16 interface) Driver: OFED Drivers v1-90-50015 Firmware: 12.14.20.36
Switch	2x Mellanox Spectrum™ SN2700 MLNX-OS® 3.6.5011

Table 2: Server Node Configuration Details

Network Infrastructure

The network is designed to provide a fully redundant interconnect fabric, ensuring no single component will interrupt data availability in production.

The network is built with two Mellanox Spectrum SN2700 Open Ethernet Switches running MLNX-OS® and providing 16x 100GbE ports each.

Physical connections are illustrated in Figure 3. Each Hyper-V/Storage Spaces Direct server has two dual-port HPE IB EDR/Ethernet 100Gb (Mellanox ConnectX®-4) network interface controllers (NICs) with a single port from each NIC connected to each switch to ensure maximum scalability can be achieved to ensure complete hardware redundancy. While it is optional to connect the second port from each server's NIC for additional redundancy, this four-connection configuration results in only 20% additional network bandwidth into/out of each host due to the limiting factor of the x16 PCIe interface being used for each NIC, restricting the overall bandwidth for each NIC (single port or dual port) to a maximum theoretical limit of 128 Gb/s. For this RA, it was decided to not connect the additional port on each NIC to reduce network switch ports consumed.

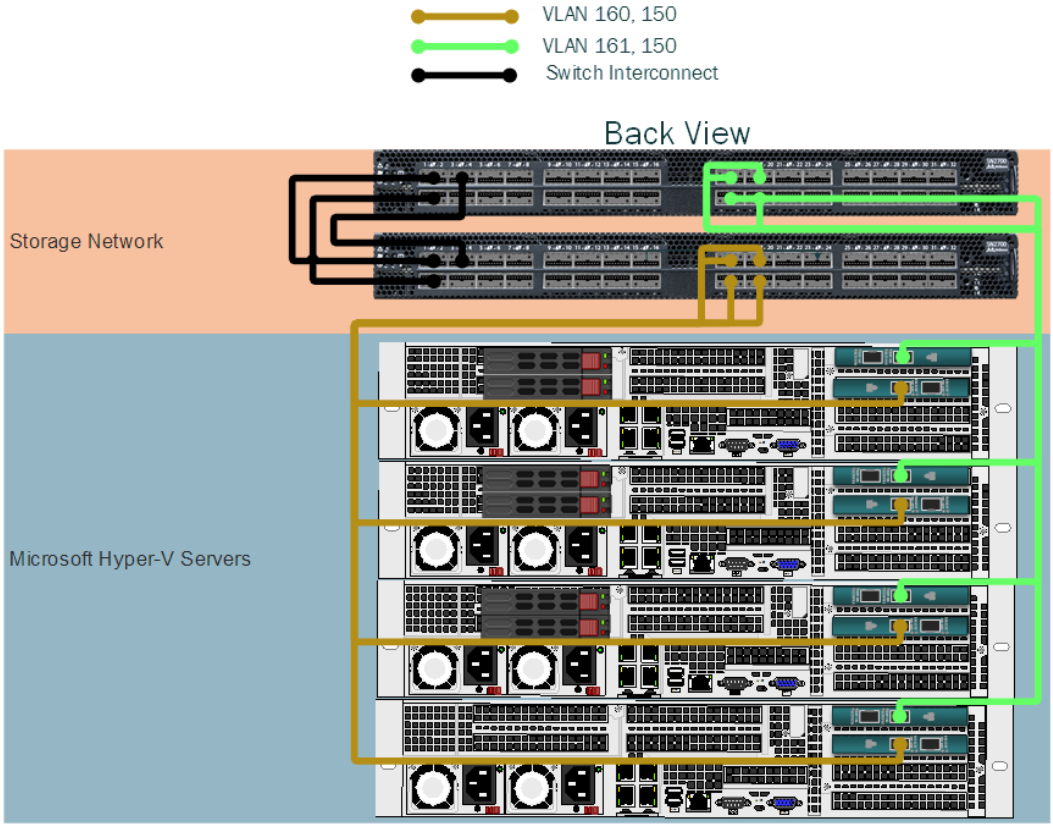


Figure 3: Hyper-V/Storage Spaces Direct Cluster Physical Layout

Each switch is interconnected using 3x 100 Gb/s ports to provide alternate data paths between application servers and storage targets.

Solution Design — Software

Operating System

All server nodes used in this solution used Microsoft Windows Server 2016 Data Center Edition version 10.0.14393. Windows Server 2016 offers advanced datacenter-class software server solutions that support either bare-metal applications or virtualized server workloads that enable multiple application services to be dynamically started, stopped, or relocated to other servers within the virtual server cluster.

For this solution, additional, optional services were installed to support an HCI solution:

- Failover-Clustering (Includes Storage Spaces Direct)
- Microsoft Windows Server Hyper-V

Hyper-V Configuration

VM Configuration

Each server node in this RA was deployed with 80 VMs. This ensured each VM accessed the resources of a single CPU core. Each VM was configured as follows:

Virtual Server Configuration	
vProcessors	2x virtual processors
vNICs	1x VM network adapter Application Client/Server Network (VLAN 150)
vRAM	4GB
Operating System	Windows Server 2016 Datacenter Core (10.0.14393)
Virtual Machine Size	24GB with 16GB datafile

Table 3: Virtual Server Configuration Details

Virtual Network Configuration

Each Hyper-V server was configured with a single virtual switch with the following configuration (Figure 4, Table 4):

Virtual Network Configuration	
VLANs	SMB-3 Data Management Network VLAN 160 VLAN 161 Application Client-Server Traffic VLAN 150
Quality of Service	SMB-3 Data Management Network: 50% Application Client Network: 50%
Virtual to Physical Affinity (VMNetworkAdapterTeamMapping)	VLAN 160: Physical NIC Port 1 VLAN 161: Physical NIC Port 2 VLAN 150: No physical port affinity (round robin)
Switch Embedded Teaming	Enabled for Physical NIC Ports

Table 4: Virtual Network Configuration Details

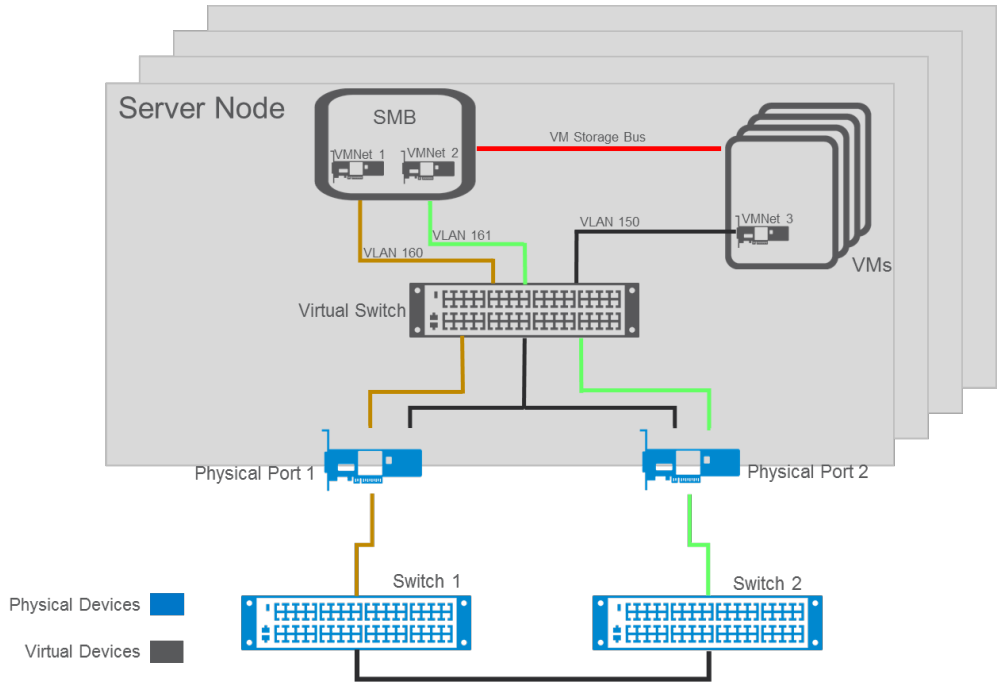


Figure 4: Virtual Server Configuration

Storage Spaces Direct Configuration

Storage Spaces Direct is configured to provide a single storage pool of 56TB raw capacity distributed across the four Hyper-V server nodes. Within this storage pool, we created four 4TB redundant virtual disks such that a single node was the virtual disk "master" for one disk each.

Within Storage Spaces Direct, virtual disks are created such that the virtual disk is hosted on every drive within the storage pool. When a write to a virtual disk is made by any VM in the cluster, that write is initially sent to that virtual disk's master. The master then writes the data to SSDs on one or more cluster nodes using the replication factor defined when creating the volume before acknowledging the write as completed to the VM.

On each virtual disk, a ReFS volume was created and a data protection configuration that would support two concurrent disk failures was used (3x replication) to ensure that node failures would not result in loss of VM access.

In addition, to support the test infrastructure used, a 400GB cluster shared volume is created to host all test logging. This volume is configured for replication factor 2.

Measuring Performance[2]

Test Methodology

Microsoft's VMFleet[3] test tool is used to deploy a fleet of 320 VMs distributed evenly across the four Hyper-V host systems (80 VMs per physical host). Within each VM, VMFleet executes a `diskspd`[4] workload and performance is measured for a series of workload definitions for all Input-Output (I/O) profiles.

Each test in the test sweep differs in terms of queue depths and write percentages for a total of 16 individual test runs per sweep. Each test was run for a specific time period:

- Small-block tests
  - Warm up: 3 minutes
  - Test run: 2 minutes
- Large-block tests
  - Warm up: 90 minutes
  - Test run: 60 minutes

Each test in the test sweep was executed a minimum of five times, with results averaged across all five test executions, to determine the reported result for each test in this RA performance results section.

To determine maximum IOPS performance, a random access 4 KiB block size was used for all I/O. The queue depths are varied from 1 to 8 at increments of a power of 2. The read-to-write ratio (R/W) varied from 100% read (100/0) to 100% write (0/100) and with two common read/write ratios (90/10, and 70/30) included, resulting in a total sweep of 16 different test data points (Table 5):

VMFleet Small Block Workload Parameters	
Block size	4 KiB
Threads /VM	1
Queue Depth /VM	1, 2, 4, 8
Write Percentages	0%, 10%, 30%, 100%

Table 5: Parameters for Small Block Workload Sweep

Large block I/O is classically associated with a several use cases such as

- DSS/BI analytics workloads that are typically large-block random reads as data is typically coming from multiple tables from a large distributed dataset.
- Data Warehouse/DSS ingest transactions that are typically large-block writes. For a single table, these writes are sequential, but if population of multiple tables is occurring, the net result is random writes to multiple tables, which becomes more random as the number of tables being populated increases.
- Video streaming and transcoding operations are typically large-block sequential read operations on a per VM basis. At the Hyper-V kernel, these per-VM sequential reads are manifested as random, large-block reads.

While there are other large-block use cases, these are most prevalent. The common requirement of these workload types is maximized throughput from the storage system.

[2] All test results were based on Hewlett Packard Enterprises DL380 Gen 10 Server configured as defined in Table 2
[3] VMFleet is a test orchestrator for deploying and executing VMs. It is a component of the diskspd storage load generator distribution.
[4] diskspd is a storage load generator and performance test tool from the Microsoft Windows Server and Cloud Server Infrastructure Engineering teams. It can be found at: https://github.com/Microsoft/diskspd

To determine maximum performance in terms of throughput in megabytes per second (MB/s), a random access 128 KiB block size was used for all I/O. The queue depths varied from 1 to 8 as a power of 2. The read/write ratios of 100% read and 100% write were used to simulate typical large-block use cases of decision support/business intelligence (100% read) and data warehouse ingest (100% write) application servers that rely on large block I/O. This results in a total sweep of eight different test data points (Table 6):

VMFleet Large Block Workload Parameters	
Block size	128 KiB
Threads/VM	1
Queue Depth /VM	1, 2, 4, 8
Write Percentages	0%, 100%

Table 6: Parameters for large block workload sweep

For each test run, various metrics were captured:

- Each VM returned the diskspd results describing the I/O as experienced within the VM (latency, throughput, and I/O operations per second)
- Each physical host captured various performance monitor metrics describing the CPU and physical disk load. Disk loads were measured at both the virtual file system (cluster shared volume) and physical device.

VMFleet was used to define and execute each of the different sweeps of the tests described above.

VMFLEET is a suite of Microsoft PowerShell scripts that greatly simplify the execution of multiple test configurations, enabling users to easily test various use cases for their solution.

Performance Results

This all-SATA SSD RA shows that Storage Spaces Direct-based HCI solutions can provide high performance at low latencies across a wide range of storage I/O profiles. Each of the sections below discuss the specific test results for both small-block IOPS (4 KiB) and large-block MB/s (128 KiB) workloads.

Small Block Size IOPS

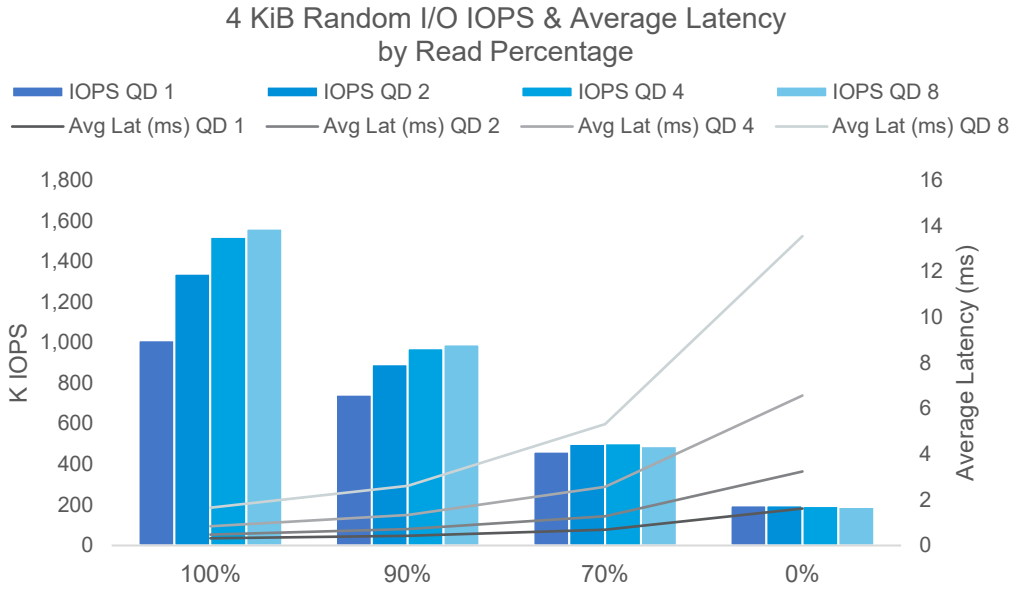


Figure 5: 4 KiB Random IOPS with Latency by Read Percentage

As the chart above shows, the 4 KiB random IOPS scaled well for all I/O profiles. Moving to a QD8 (QD640 at the Hyper-V kernel layer) resulted in a slight drop in overall performance as write percentage increased in terms of IOPS, with a significant increase in overall latency as write percentage increased.

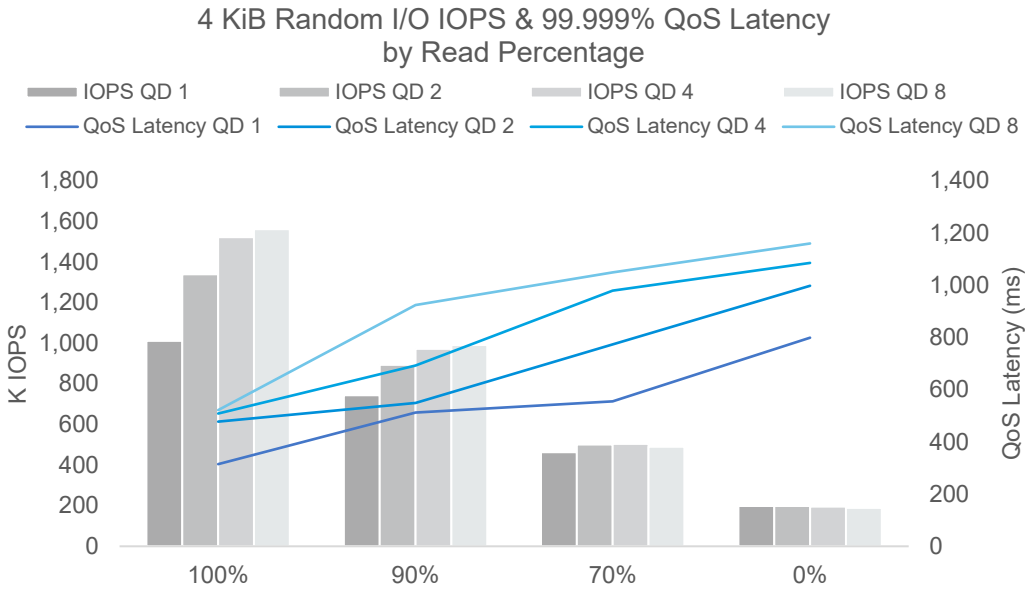


Figure 6: 4 KiB Random IOPS with QoS Latency by Read Percentage

Looking at 99.999% QoS latency, as expected, tail latency increases steadily as we move from 100% reads to 100% writes; however, as we reach maximum I/O performance at differing queue depths as read/write ratio changes, we see a steady increase in tail latency, with increasing queue depth at each I/O profile with reduced increases in overall IOPS at higher write percentages. For a typical 70/30 read/write profile, QD2 is the most optimal configuration providing almost identical throughput with 33% lower tail latency of QD4.

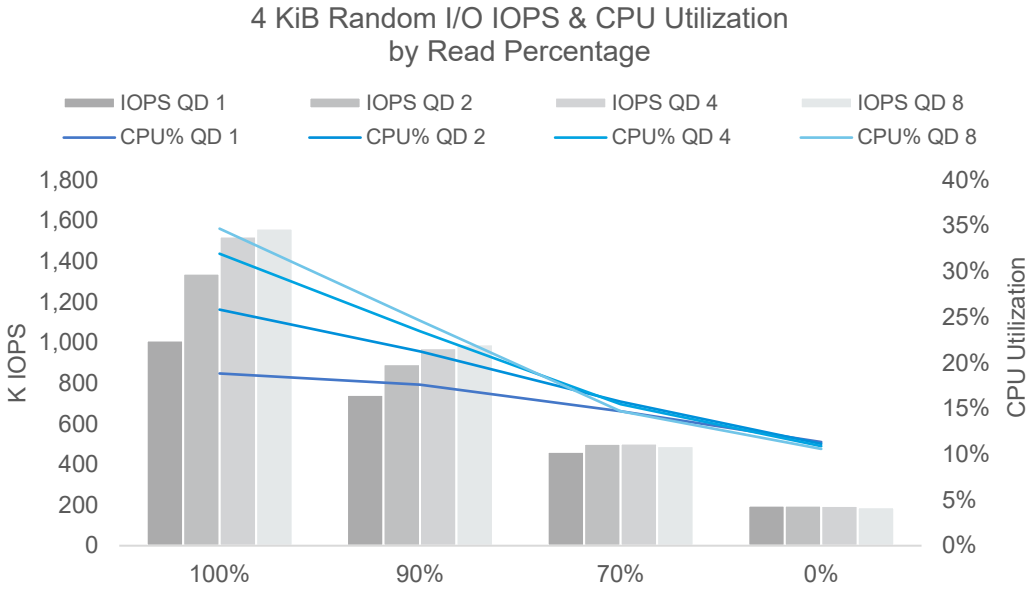


Figure 7: 4 KiB Random IOPS with CPU Utilization by Read Percentage

Finally, looking at CPU utilization, higher read percentages require higher CPU time, and higher queue depths at each I/O profile require higher CPU time; however, deeper analysis shows that as the read percentage decreases (write percentage increases), the performance of the QD1 configuration gets closer to the QD2 or QD3 configuration. The same behavior is observed in the CPU utilization. As write percentage increases, the CPU utilization of QD1 gets to nearly the same value as QD2 and QD3.

The table below summarizes the small block I/O test results

Read %		100%	90%	70%	0%
IOPS	QD 1	1,010,827	743,031	462,559	197,965
	QD 2	1,339,248	893,399	501,168	197,789
	QD 4	1,521,334	971,560	503,299	195,262
	QD 8	1,561,441	990,902	490,139	189,533
Latency(ms)	QD 1	0.32	0.43	0.69	1.62
	QD 2	0.48	0.72	1.28	3.24
	QD 4	0.84	1.33	2.56	6.57
	QD 8	1.66	2.62	5.32	13.55
99.999% Latency(ms)	QD 1	314.55	511.91	555.05	798.67
	QD 2	477.33	548.53	772.59	997.46
	QD 4	508.32	691.60	979.07	1085.01
	QD 8	521.49	923.81	1048.83	1159.40
CPU Utilization	QD 1	18.8%	17.6%	14.7%	11.3%
	QD 2	25.8%	21.2%	15.8%	11.1%
	QD 4	31.9%	23.5%	15.5%	10.9%
	QD 8	34.7%	24.6%	14.7%	10.6%

Table 7: Summary of 4 KiB Random Performance

Large Block Size Throughput

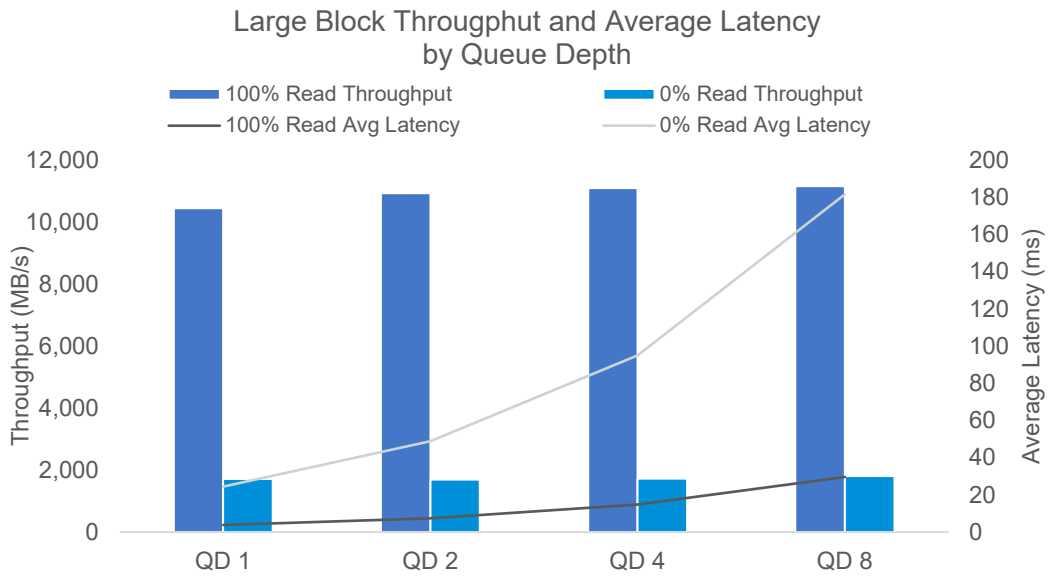


Figure 8: Large Block Throughput and Average Latency

Large-block read I/O shows small increases in overall throughput as the queue depth increases. The accompanying increase in average latency for reads increases consistently and significantly as we increase queue depth, with I/O performance improving by 11% from QD1 to QD8, while latency increases by 670% indicating that—for heavy read-centric solutions—there is no value of increased queue depth within the Hyper-V cluster.

Large-block write throughput shows there are small increases as the queue depth increases (5%), while the latency increases substantially (640%). Again, for large block solutions, lower queue depths provide the most efficient performance.

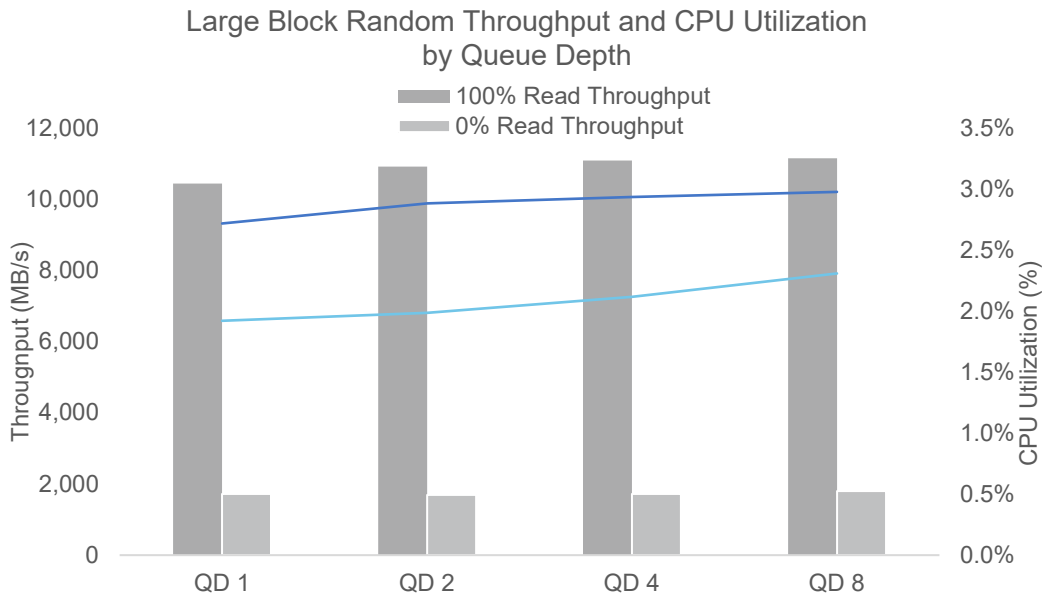


Figure 9: Large Block Throughput and CPU Utilization

Reviewing CPU utilization shows that it remains quite low (1.9% to 3%) for all tested configurations. When viewed in conjunction with the 4k random CPU data from earlier, the conclusion is that CPU is related to number of I/O operations rather than raw throughput. The table below summarizes the large-block throughput data:

Read %		100%	0%
Throughput	QD 1	10,444	1,719
	QD 2	10,927	1,694
	QD 4	11,094	1,722
	QD 8	11,159	1,806
Average Latency(ms)	QD 1	3.8	24.5
	QD 2	7.4	48.9
	QD 4	14.8	95.1
	QD 8	29.6	181.6
CPU Utilization	QD 1	2.7%	1.9%
	QD 2	2.9%	2.0%
	QD 4	2.9%	2.1%
	QD 8	3.0%	2.3%

Table 8: Summary of 4 KiB Random Performance

Queue Depth

The workloads described in this paper operate by maintaining a specified queue depth inside each VM. The next chart shows how QD in each VM across 320 VMs translates to the average queue length for each physical SSD.

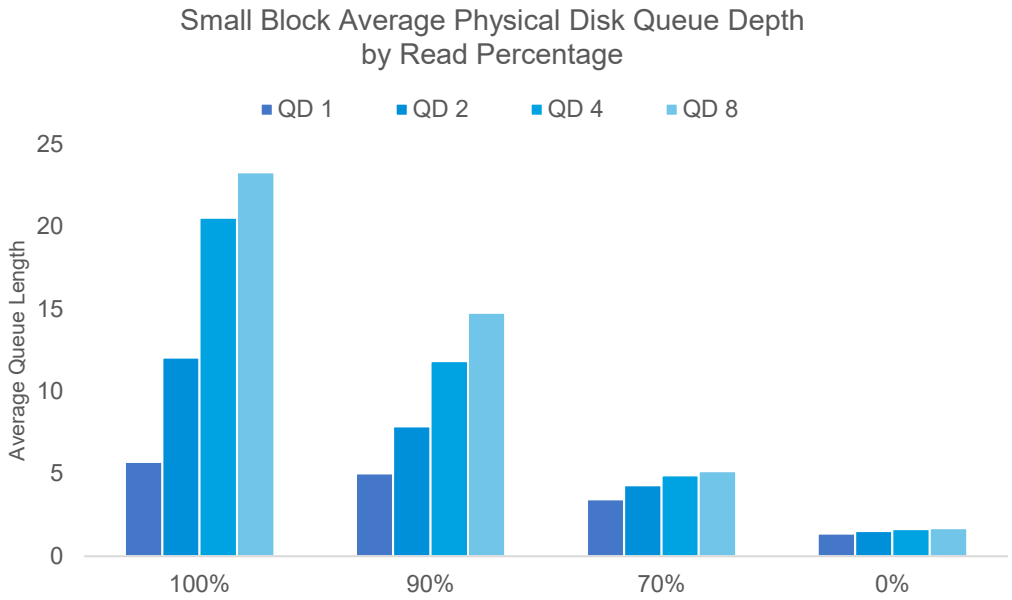


Figure 10: Small Block Physical Disk Queue Length Analysis

Read Percentage	QD 1	QD 2	QD 4	QD 8
100%	5.7	12.1	20.5	23.3
90%	5.0	7.9	11.8	14.8
70%	3.5	4.3	4.9	5.2
0%	1.4	1.5	1.6	1.7

Table 9: Per SSD Average Queue Length for Small Block I/O

If the only latency in the system is due to the physical drive, then the queue depth would be predictable based on the VM's queue depth and the number of VMs and physical devices:

$$SSD\ Queue\ Length = \frac{320\ VMs * QD_{VM}}{32\ Drives}$$

For each random workload, the average queue length on each physical SSD is less than 8X the VM queue depth (at QD_{VM}=8). This suggests that the Storage Spaces Direct overhead adds some additional latency to each I/O operation.

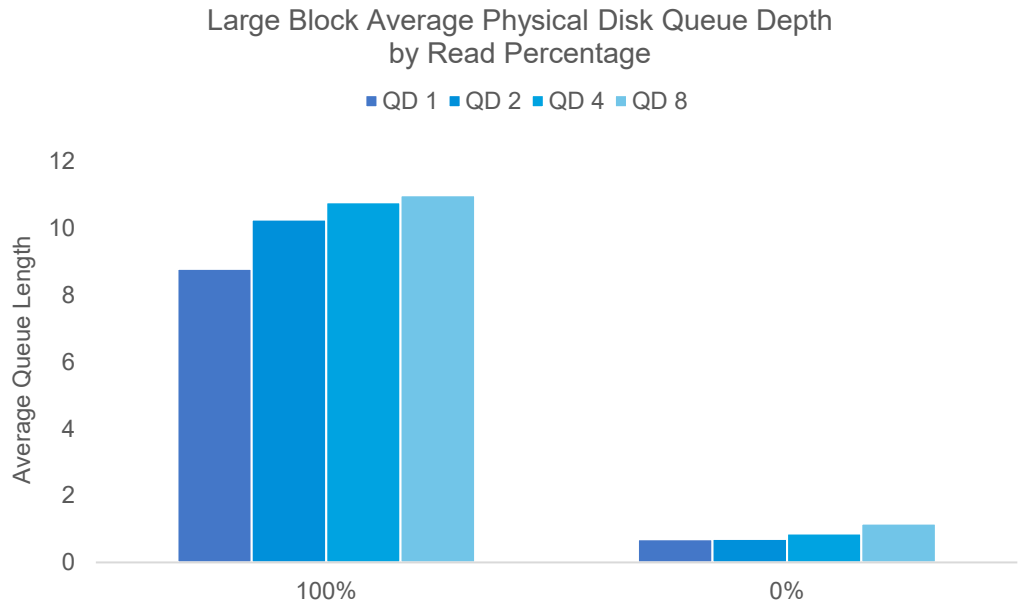


Figure 11: Per SSD Queue Length for Large Block I/O

Read Percentage	QD 1	QD 2	QD 4	QD 8
100%	8.8	10.3	10.8	11.0
0%	0.7	0.7	0.9	1.2

Table 10: Per SSD Queue Length for Large Block I/O

The story is slightly different for large block reads. Here, the queue length on the physical devices at QD1 and QD2 exceed 8X the VM QD. This suggests that configuration performance is limited by the physical devices.

Endurance

The Micron 5200 ECO 1920GB SSD used in this RA has a rated lifetime endurance, for warranty purposes, of 3500TB total bytes written (TBW) over a 5-year period. This translates to a constant sustainable throughput rate of 22.2 MB/sec while still staying within this TBW:

$$3{,}500{,}000{,}000MB * \frac{1}{5\ Years} * \frac{1\ Year}{365\ Days} * \frac{1\ Day}{24\ Hours} * \frac{1\ Hour}{3600\ Seconds} = 22.196\ {}^{MB}/_{Sec}$$

For workloads that are time dependent, we can multiply by 2 or 3 to get the results for 8 or 12 hours of workload:

Sustainable Write Rate by Hours per Day	
24 Hours / Day	22.2 MB/s
12 Hours / Day	44.4 MB/s
8 Hours / Day	66.6 MB/s

Table 11: Endurance for Different Solution Duty Cycles

For the workloads tested, the writes to disk are shown in the figure below. As the graph shows, most workloads—when running at maximum performance levels that were attained—would exceed drive warranty.

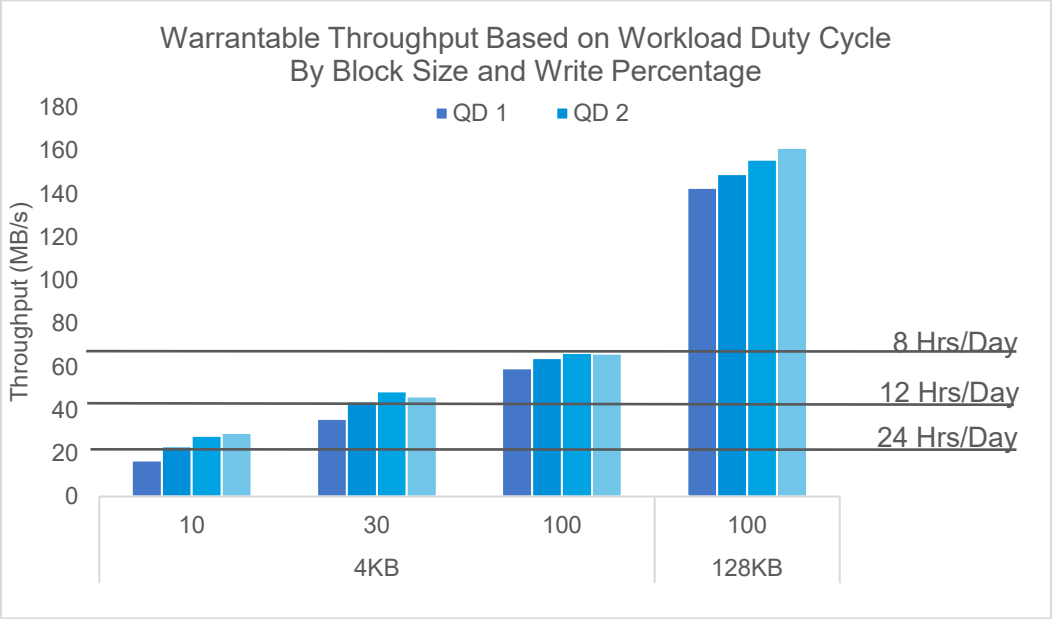


Figure 12: Warrantable Throughput Based on Workload Duty Cycle

If the target workload is heavily write-oriented, there are two options that can allow staying within warranty limits:

- Moving to higher capacity ECO drives will increase endurance. The 3960GB 5200 ECO has 2X the capacity and 2X the endurance of the 1920GB drives used in this RA.
- Moving from read-intensive ECO drives to higher endurance alternatives such as the 5200 PRO or 5200 MAX SSD. For example, the 1920TB 5200 PRO has an endurance of 5950 total TB written and the 1920 MAX has an endurance of 17520 total TB written over 5 years.

The graph below compares these endurances.

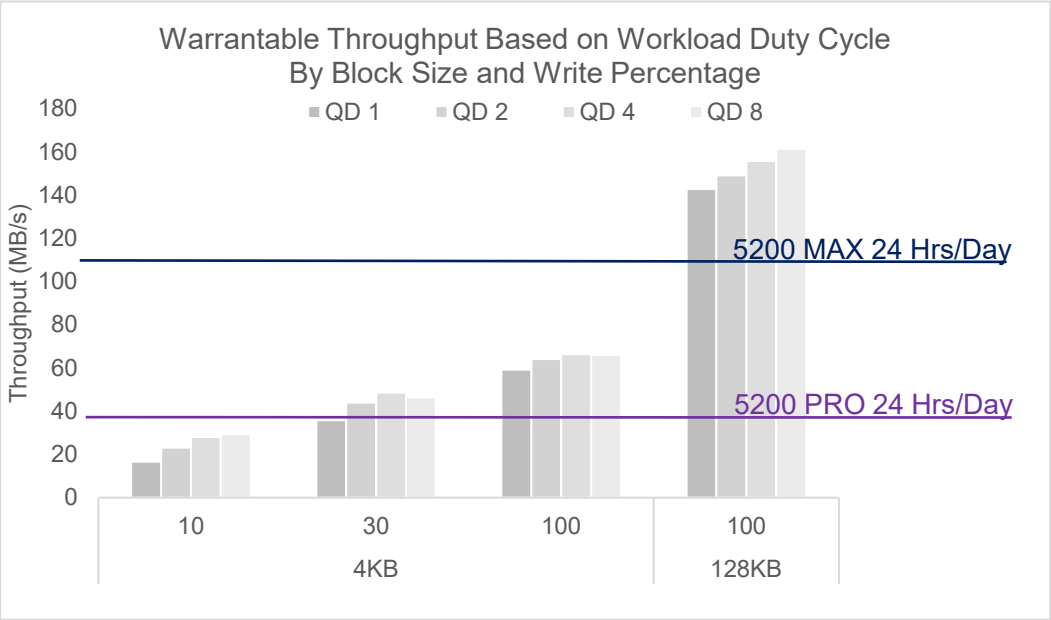


Figure 13: 5200 PRO and 5200 MAX Endurance Comparisons

Understanding the specific needs of your environment will determine eventual durability of the SSD solution. High write workloads will require utilizing higher endurance options.

Impact of Storage Spaces Direct Writes on SSDs

Based on the selected Storage Spaces Direct data protection, when using three copies of all written data (Replication factor 3; sometimes called application write amplification), a best-case expectation would be that every write requested by a VM would result in exactly three writes. This replication factor is expected to be the largest contributor to write traffic. Other sources are the metadata, indexes, etc. associated with managing a distributed storage system. The analysis, shown in Figure 14, illustrates there is almost no additional overhead added by the file-system or Storage Spaces Direct.

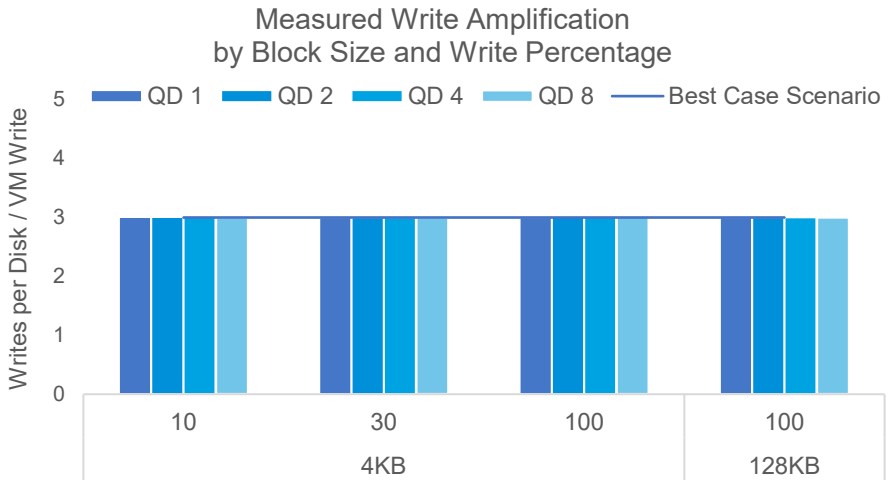


Figure 14: Write Amplification for Storage Spaces Direct

Write Percentage		QD 1	QD 2	QD 4	QD 8
Small Block	10%	3.01	3.01	3.01	3.01
	30%	3.00	3.00	3.00	3.00
	100%	3.01	3.02	3.01	3.02
Large Block	100%	3.00	3.00	3.00	3.00

Table 12: Write Amplification for Storage Spaces Direct

For all workload profiles tested, there is no more than 2% of additional write amplification over that required for data protection when using Storage Spaces replication, indicating that sum of all VM write throughput measurements multiplied by the pool's replication factor can be used to size the total storage solution and be used for lifetime predictions of the SSDs deployed. It should be noted that we have not done this same analysis of erasure coding configurations as part of this RA, and this recommendation should not be assumed to hold for erasure coded solutions.

Additional Planning Considerations

- **Minimum number of nodes:** Microsoft Storage Spaces Direct storage software is architected as a shared-nothing, scale-out cluster. To ensure all data is redundant and that the cluster health can be assured, at least three servers must be deployed. This ensures high availability may be maintained in the event of a node loss or communications issue on the network. Three nodes ensure that high-availability can be maintained and that server and/or communications issues with any single node can be identified through a quorum mechanism.
- **Data redundancy:** Data protection and availability is a base requirement for any enterprise-class HCI solution. Data redundancy ensures any data stored within the Storage Spaces Direct storage repository is copied to two or more nodes. This replication factor protects from any node failure or communications interruptions between nodes. Higher replication factors equate to a higher number of simultaneous node failures that can occur before cluster failure. This RA uses replication equal to 3—three copies of any data is stored, each on a different cluster node—for all test data, ensuring the cluster can accept two node failures.
- **Alternative server specifications:** Depending on the number of VMs being deployed and the workload running on each VM, it may be necessary to use higher (or lower) performance-level CPU options or that of a completely different CPU architecture—such as AMD's EPYC architecture—to create even better performance and cost-optimized deployments.
- **Server vendor:** This RA does not require Hyper-V/Storage Spaces Direct servers from any specific server vendor. We strive to focus on the architectural requirements for each solution discussed, allowing you to select the server vendor of your choice.

Appendix A: Cluster Configuration PowerShell Script

```
$DebugPreference = "Continue"
$start_datetime = Get-Date

$dom_hosts = @("G10-01.s2d.lcl", "G10-02.s2d.lcl", "G10-03.s2d.lcl", "G10-04.s2d.lcl")
$hosts = @("172.16.6.3", "172.16.9.122", "172.16.9.120", "172.16.9.121")
$hostnames = @("G10-01", "G10-02", "G10-03", "G10-04")

# I use a weird way to assign ip addresses. I didn't like any of the
# IP address plugins for powershell so I use a basic array and just
# increment. I need to keep track of how many hosts I have and if I can
# increment the relevent IP address portion that many times and stay
# in the subnet. 4 hosts makes this fairly easy.
$smb1_network = @(192,168,32,128)
$smb2_network = @(192,168,32,160)
$smb_mask = 27


$dom_user = "s2d\Administrator"
$dom_pass = "SomePassw0rd"
$dom = "s2d.lcl"
$dom_ip_prefix = "172.17.3*"
$dom_secure_string = New-Object -TypeName System.Security.SecureString
$dom_pass.ToCharArray() | ForEach-Object {$dom_secure_string.AppendChar($_)}
$dom_cred = New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList
$dom_user,$dom_secure_string

$user = "Administrator"
$pass = "SomePassw0rd$"
$secure_string = New-Object -TypeName System.Security.SecureString
$pass.ToCharArray() | ForEach-Object {$secure_string.AppendChar($_)}
$cred = New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList
$user,$secure_string


Write-Output "Start time is $start_datetime"

# Set firewall to disabled
# Rename the hosts based on the hostnames array
# Install prerequisite packages:
#   Failover-Clustering
#   Hyper-V
#   Data center Bridging
foreach($h in $hosts)  {
    $session = New-PSSession -ComputerName $h -Credential $cred
    $newname = $hostnames[$hosts.IndexOf($h)]
    Write-Output "Setting Firewall on host: $h"
    Write-Output "Renaming host $h to $newname"
    Invoke-Command -session $session -ArgumentList $newname {
        # Set firewall wide open and enable winrm usage so my automation
        # works with little hassle
        # DO NOT DO THIS IN PRODUCTION
        Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled False
        Rename-Computer -NewName $args[0]
        Set-TimeZone -Name "Central Standard Time"
        winrm set winrm/config/service '@{AllowUnencrypted="true"}'
        winrm set winrm/config/service/auth '@{Basic="true"}'
    }

    # Copying drivers and tools
    Copy-Item -Path C:\Users\public\Downloads\MLNX_WinOF2-1_70_All_x64.exe -Destination c:\ -
```

```
ToSession $session
    Copy-Item -Path C:\Users\public\Downloads\msecli_Windows_64bit.zip -Destination c:\ -
ToSession $session
    Copy-item -Path C:\Users\public\Downloads\python-2.7.14.amd64.msi -Destination C:\ -
ToSession $session

    Write-Output "Installing Clustering, Hyper-V and DCB on host: $h"
    Invoke-Command -session $session {
        Install-WindowsFeature -Name Failover-Clustering -IncludeManagementTools -
IncludeAllSubFeature
        Install-WindowsFeature -Name Hyper-V -IncludeManagementTools -IncludeAllSubFeature
        Install-WindowsFeature -Name Data-Center-Bridging

        # Installing OFED drivers for Mellanox NICs
        c:\MLNX_WinOF2-1_70_All_x64.exe /S /v" /qn"

        # Installing python for some internal tools
        msiexec /i c:\python-2.7.14.amd64.msi /passive /norestart ADDLOCAL=ALL

        # Installing msecli for physical device management
        Expand-Archive 'C:\msecli_windows_64bit.zip' -DestinationPath c:\msecli
    }
}

# Restart the hosts for installs and rename to take effect
foreach($h in $hosts) {
    Write-Output "Restarting host: $h"
    Restart-Computer -ComputerName $h -Credential $cred -Force -wait
}

# Configure the SET switches and RDMA settings
foreach($h in $hosts) {
    $session = New-PSSession -ComputerName $h -Credential $cred
    Invoke-Command -session $session {
        # In my configuration, I tested a -like string to capture my specific
        # Mellanox cards. This was slightly difficult as I have 10Gb Mellanox
        # onboard that I do not want to use for this architecture. Test your
        # specific environment before running any scripts to ensure that you
        # are capturing the correct adapters
        $RDMAAdapters = Get-NetAdapter | ? {$_.InterfaceDescription -like "*100Gb*"}

        # Enable QoS Flow Control
        # The priority number for SMB needs to match the switch configuration.
        # But the specific value is not important
        New-NetQosPolicy "SMB" -NetDirectPortMatchCondition 445 -PriorityValue8021Action 3
        Enable-NetQosFlowControl -Priority 3
        Disable-NetQosFlowControl -Priority 0,1,2,4,5,6,7
        Enable-NetAdapterQos -InterfaceDescription $RDMAAdapters.InterfaceDescription
        # Only 50% of the network bandwidth is allocated to SMB. This ensures
        # that the client networks have enough bandwidth for normal operation.
        New-NetQosTrafficClass "SMB" -Priority 3 -BandwidthPercentage 50 -Algorithm ETS

        # Set flow control properties on the net adapters
        foreach($adapter in $RDMAAdapters) {Set-NetAdapterAdvancedProperty -
InterfaceDescription $adapter.InterfaceDescription -RegistryKeyword "*FlowControl" -
RegistryValue 0}
        foreach($adapter in $RDMAAdapters) {Set-NetAdapterAdvancedProperty -
InterfaceDescription $adapter.InterfaceDescription -RegistryKeyword "*JumboPacket" -
RegistryValue 9014}

        # Creat the SETSwitch with SMB and Client network adapters
        New-VMSwitch -Name 100G -NetAdapterInterfaceDescription
$RDMAAdapters.InterfaceDescription -EnableEmbeddedTeaming $true -AllowManagementOS $false
```

```
        Add-VMNetworkAdapter -SwitchName 100G -Name SMB1 -ManagementOS
        Add-VMNetworkAdapter -SwitchName 100G -Name SMB2 -ManagementOS
        Add-VMNetworkAdapter -SwitchName 100G -Name s2d -ManagementOS

        # Enable RDMA on the SMB net adapters
        $SMBAdapters = Get-NetAdapter | ? {$_.Name -like "*SMB*"}
        Enable-NetAdapterRDMA -Name $SMBAdapters.Nam
        foreach($adapter in $SMBAdapters) {Set-NetAdapterAdvancedProperty -
InterfaceDescription $adapter.InterfaceDescription -RegistryKeyword "*JumboPacket" -
RegistryValue 9014}

        # Assign VLANs to each net adapter
        Set-VMNetworkAdapterVlan -VMNetworkAdapterName s2d -VlanId 150 -Access -ManagementOS
        Set-VMNetworkAdapterVlan -VMNetworkAdapterName SMB1 -VlanId 160 -Access -ManagementOS
        Set-VMNetworkAdapterVlan -VMNetworkAdapterName SMB2 -VlanId 161 -Access -ManagementOS
    }
}

# Configure NIC affinity. We need a 1:1 mapping of physical to virtual ports.
# In this case, I match port 1 to SMB1 and port 2 to SMB2 where only one of
# each port is connected and UP. (port 1 in card 1 is SMB1 and port 2 on
# card 2 is SMB2. Ports 2 on card 1 and port 1 on card 2 are not used)
foreach($h in $hosts) {
    $session = New-PSSession -ComputerName $h -Credential $cred
    write-output "Setting nic affinity for host $h"
    Invoke-Command -Session $session {
        $nics1 = Get-NetAdapter -Name "*Port 1" | ? {$_.Status -eq "Up"}
        $nics2 = Get-NetAdapter -Name "*Port 2" | ? {$_.Status -eq "Up"}
        $nics1.Name | % {set-vmnetworkadapterteammapping -VMNetworkAdapterName "SMB1" -
ManagementOS -PhysicalNetAdapterName $_}
        $nics2.Name | % {set-vmnetworkadapterteammapping -VMNetworkAdapterName "SMB2" -
ManagementOS -PhysicalNetAdapterName $_}
    }
}

# Add the hosts to the domain
# Disable the initial dhcp interface
# Restart hosts
foreach($h in $hosts) {
    Write-Output "Adding host $h to domain $dom"
    $session = New-PSSession -ComputerName $h -Credential $cred
    Invoke-Command -Session $session -ArgumentList $dom, $dom_cred {
        Add-Computer -DomainName $args[0] -Credential $args[1]
    }

    $dom_netips = Invoke-Command -session $session {
        Get-NetIPAddress | ? {$_.AddressFamily -like "IPv4"}
    }

    # The interface we're connecting to the nodes with is a 1Gb DHCP interface. We
    # will disable this interface and use the virtual network adapter we created
    # above. If we're going to disable the interface that we're using to connect
    # to the host we should get the IP for the new interface and use a connection
    # through that instead. So we do. Get the other IP on the domain network and
    # create a new session we used to send the commands to shutdown the first
    # interface.
    $new_ip = $dom_netips | ? {$_.IPAddress -like "{0}*" -f $dom_ip_prefix}
    $new_ip = $new_ip.IPAddress

    Write-Output "New ip for host: $h is: $new_ip"
    $new_session = New-PSSession -ComputerName $new_ip -Credential $cred
    Invoke-Command -session $new_session -ArgumentList $h {
        $lab_ip = Get-NetIPAddress -IPAddress $args[0]
```

```
        Get-NetAdapter -Name $lab_ip.InterfaceAlias | Disable-NetAdapter -Confirm:$false
        Get-NetAdapter | ? {$_.Status -like "Disconnected"} | Disable-NetAdapter -
Confirm:$false
    }

    Write-Output "Restarting host: $h"
    Restart-Computer -ComputerName $new_ip -Credential $cred -Force -Wait
}

# Set IP addresses for the SMB interfaces
foreach($h in $dom_hosts) {
    $session = New-PSSession -ComputerName $h -Credential $dom_cred
    $incr = $dom_hosts.IndexOf($h) + 1

    # We generate the interfaces through simple addition to the base values
    # of the arrays
    $smb1_ip = ($smb1_network[0],$smb1_network[1],$smb1_network[2],($smb1_network[3] +
$incr)) -join "."
    $smb2_ip = ($smb2_network[0],$smb2_network[1],$smb2_network[2],($smb2_network[3] +
$incr)) -join "."
    Write-Output "Interfaces - $smb1_ip - $smb2_ip - $smb_mask"
    Invoke-Command -Session $session -ArgumentList $smb1_ip, $smb2_ip, $smb_mask {
        Get-NetAdapter | ? {$_.Name -like "*SMB*"} | Remove-NetIPAddress -AddressFamily ipv4
-Confirm:$false

        $smb1 = Get-NetAdapter | ? {$_.Name -like "*SMB1*"}
        $smb2 = Get-NetAdapter | ? {$_.Name -like "*SMB2*"}
        $ip1 = [IPAddress] $args[0]
        $ip2 = [IPAddress] $args[1]

        Set-NetIPInterface -InterfaceAlias $smb1.InterfaceAlias -Dhcp Disabled
        Set-NetIPInterface -InterfaceAlias $smb2.InterfaceAlias -Dhcp Disabled
        New-NetIPAddress -InterfaceAlias $smb1.InterfaceAlias -IPAddress
$ip1.IPAddressToString -PrefixLength $args[2]
        New-NetIPAddress -InterfaceAlias $smb2.InterfaceAlias -IPAddress
$ip2.IPAddressToString -PrefixLength $args[2]
    }

    Write-Output "Restarting host: $h"
    Restart-Computer -ComputerName $h -Credential $dom_cred -Force
}

# At the end, the hosts will still be rebooting as we dropped
# the -wait pararmeter from Restart-Computer
$stop_datetime = Get-Date
Write-Output "Started at $start_datetime"
Write-Output "Stopped at $stop_datetime"
```

Appendix B: Mellanox SN2100 Configurations

Switch01

```
##
## MLAG protocol
##
   protocol mlag

##
## Interface Ethernet configuration
##
   interface ethernet 1/17 mtu 9216 force
   interface ethernet 1/18 mtu 9216 force
   interface ethernet 1/19 mtu 9216 force
   interface ethernet 1/20 mtu 9216 force
   interface ethernet 1/32 mtu 9216 force

   interface mlag-port-channel 1
   interface mlag-port-channel 1 mtu 9216 force
   interface port-channel 1

   interface ethernet 1/1 channel-group 1 mode active
   interface ethernet 1/2 channel-group 1 mode active
   interface ethernet 1/3 channel-group 1 mode active

   interface ethernet 1/17 switchport mode trunk
   interface ethernet 1/18 switchport mode trunk
   interface ethernet 1/19 switchport mode trunk
   interface ethernet 1/20 switchport mode trunk

   interface ethernet 1/32 mlag-channel-group 1 mode active
   interface mlag-port-channel 1 switchport mode trunk
   interface mlag-port-channel 1 no shutdown

##
## VLAN configuration
##
   vlan 2-501
   vlan 502-1001
   vlan 1002-1501
   vlan 1502-2001
   vlan 2002-2048
   vlan 4001


##
## STP configuration
##
no spanning-tree

##
## LAG configuration
##
   lacp

##
## L3 configuration
##
   ip routing vrf default
   interface vlan 4001
   interface vlan 4001 ip address 192.168.33.1 255.255.255.0
```

```
##
## DCBX PFC configuration
##
   dcb priority-flow-control enable force
   dcb priority-flow-control priority 3 enable
   interface ethernet 1/1 dcb priority-flow-control mode on force
   interface ethernet 1/2 dcb priority-flow-control mode on force
   interface ethernet 1/3 dcb priority-flow-control mode on force

   interface ethernet 1/17 dcb priority-flow-control mode on force
   interface ethernet 1/18 dcb priority-flow-control mode on force
   interface ethernet 1/19 dcb priority-flow-control mode on force
   interface ethernet 1/20 dcb priority-flow-control mode on force

   interface port-channel 1 dcb priority-flow-control mode on force

##
## LLDP configuration
##
   lldp

##
## QoS switch configuration
##

   interface ethernet 1/17 traffic-class 3 congestion-control ecn minimum-relative 20
maximum-relative 80
   interface ethernet 1/18 traffic-class 3 congestion-control ecn minimum-relative 20
maximum-relative 80
   interface ethernet 1/19 traffic-class 3 congestion-control ecn minimum-relative 20
maximum-relative 80
   interface ethernet 1/20 traffic-class 3 congestion-control ecn minimum-relative 20
maximum-relative 80

##
## MLAG configurations
##
   mlag-vip infra15-mlag100 ip 172.16.2.155 /17 force
no mlag shutdown
   mlag system-mac 00:00:5E:00:01:5D
   interface port-channel 1 ipl 1
   interface vlan 4001 ipl 1 peer-address 192.168.33.2
```

Switch02

```
##
## MLAG protocol
##
   protocol mlag

##
## Interface Ethernet configuration
##
   interface ethernet 1/17 mtu 9216 force
   interface ethernet 1/18 mtu 9216 force
   interface ethernet 1/19 mtu 9216 force
   interface ethernet 1/20 mtu 9216 force
   interface ethernet 1/32 mtu 9216 force

   interface mlag-port-channel 1
   interface mlag-port-channel 1 mtu 9216 force
   interface port-channel 1
```

```
   interface ethernet 1/1 channel-group 1 mode active
   interface ethernet 1/2 channel-group 1 mode active
   interface ethernet 1/3 channel-group 1 mode active

   interface ethernet 1/17 switchport mode trunk
   interface ethernet 1/18 switchport mode trunk
   interface ethernet 1/19 switchport mode trunk
   interface ethernet 1/20 switchport mode trunk

   interface ethernet 1/32 mlag-channel-group 1 mode active
   interface mlag-port-channel 1 switchport mode trunk
   interface mlag-port-channel 1 no shutdown

##
## VLAN configuration
##
   vlan 2-501
   vlan 502-1001
   vlan 1002-1501
   vlan 1502-2001
   vlan 2002-2048
   vlan 4001


##
## STP configuration
##
no spanning-tree

##
## LAG configuration
##
   lacp

##
## L3 configuration
##
   ip routing vrf default
   interface vlan 4001
   interface vlan 4001 ip address 192.168.33.2 255.255.255.0

##
## DCBX PFC configuration
##
   dcb priority-flow-control enable force
   dcb priority-flow-control priority 3 enable
   interface ethernet 1/1 dcb priority-flow-control mode on force
   interface ethernet 1/2 dcb priority-flow-control mode on force
   interface ethernet 1/3 dcb priority-flow-control mode on force

   interface ethernet 1/17 dcb priority-flow-control mode on force
   interface ethernet 1/18 dcb priority-flow-control mode on force
   interface ethernet 1/19 dcb priority-flow-control mode on force
   interface ethernet 1/20 dcb priority-flow-control mode on force

   interface port-channel 1 dcb priority-flow-control mode on force

##
## LLDP configuration
##
   lldp
```

```
##
## QoS switch configuration
##

   interface ethernet 1/17 traffic-class 3 congestion-control ecn minimum-relative 20
maximum-relative 80
   interface ethernet 1/18 traffic-class 3 congestion-control ecn minimum-relative 20
maximum-relative 80
   interface ethernet 1/19 traffic-class 3 congestion-control ecn minimum-relative 20
maximum-relative 80
   interface ethernet 1/20 traffic-class 3 congestion-control ecn minimum-relative 20
maximum-relative 80

##
## MLAG configurations
##
   mlag-vip infra15-mlag100 ip 172.16.2.155 /17 force
no mlag shutdown
   mlag system-mac 00:00:5E:00:01:5D
   interface port-channel 1 ipl 1
   interface vlan 4001 ipl 1 peer-address 192.168.33.1
```

Appendix C: About

Micron

Micron Technology (Nasdaq: MU) is a world leader in innovative memory solutions. Through our global brands—Micron, Crucial® and Ballistix®—our broad portfolio of high-performance memory technologies, including DRAM, NAND, NOR Flash and 3D XPoint™ memory, is transforming how the world uses information. Backed by more than 35 years of technology leadership, Micron's memory solutions enable the world's most innovative computing, consumer, enterprise storage, data center, mobile, embedded, and automotive applications. Micron's common stock is traded on the Nasdaq under the MU symbol. To learn more about Micron Technology, Inc., visit micron.com.

Revision History

Rev. A	07/2018	Initial release based on Micron 5100 ECO
Rev. B	09/2018	Updated for release of Micron 5200 ECO

micron.com

Rev. B 9/18 CCM004-676576390-11088