v4.0 | April 2023 | BP-2015

BEST PRACTICES

Microsoft SQL Server on Nutanix

NUTANIX
YOUR ENTERPRISE CLOUD

Copyright

Contents

1. Executive Summary

This document makes recommendations for designing, optimizing, and scaling Microsoft SQL Server deployments on the Nutanix Cloud Platform. Historically, it has been a challenge to virtualize SQL Server because of the high cost of traditional virtualization stacks and the impact that a SAN-based architecture can have on performance. Businesses and their IT departments have constantly fought to balance cost, operational simplicity, and consistent predictable performance.

Nutanix removes many of these challenges and makes virtualizing a business-critical application such as SQL Server much easier. Nutanix storage is a software-defined solution that provides all the features one typically expects in an enterprise SAN, without a SAN's physical limitations and bottlenecks. SQL Server particularly benefits from the following storage features:

- Localized I/O and the use of flash for index and key database files to lower operation latency.
- A highly distributed approach that can handle both random and sequential workloads.
- The ability to add new nodes and scale the infrastructure without system downtime or performance impact.
- Nutanix data protection and disaster recovery workflows that simplify backup operations and business continuity processes.

Nutanix lets you run both Microsoft SQL Server and other VM workloads simultaneously on the same platform. Density for SQL Server deployments is driven by the database's CPU and storage requirements. To take full advantage of the system's performance and capabilities, validated testing shows that it's better to scale out and increase the number of SQL Server VMs on the Nutanix platform than to scale up individual SQL Server instances. The Nutanix platform handles SQL Server's demanding throughput and transaction requirements with localized I/O, server-attached flash, and distributed data protection capabilities.

2. Introduction

Audience

This best practice document is part of the Nutanix Solutions Library. We wrote it for those architecting, designing, managing, and supporting Nutanix infrastructures. Readers should already be familiar with a hypervisor (VMware vSphere, Microsoft Hyper-V, or the native Nutanix hypervisor, AHV), Microsoft SQL Server, and Nutanix.

The document addresses key items for each role, enabling a successful design, implementation, and transition to operation. Most of the recommendations apply equally to all currently supported versions of Microsoft SQL Server. We call out differences between versions as needed.

Purpose

This document covers the following topics:

- Overview of the Nutanix solution
- The benefits of running Microsoft SQL Server on Nutanix
- Overview of high-level SQL Server best practices for Nutanix
- Design and configuration considerations when architecting a SQL Server solution on Nutanix
- Virtualization optimizations for VMware ESXi, Microsoft Hyper-V, and Nutanix AHV

Document Version History

Version Number	Published	Notes
1.0	April 2015	Original publication.

Version Number	Published	Notes
2.0	September 2016	Updated for SQL Server 2016 and added discussion of Nutanix Volumes.
2.1	May 2017	Updated for all-flash platform availability.
3.0	August 2019	Major updates throughout.
3.1	November 2019	Updated the Understanding RAM Configuration section.
3.2	June 2020	Updated the Nutanix overview and the Use of Jumbo Frames section.
3.3	October 2020	Updated the Best Practices Checklist section.
3.4	December 2020	Updated the SQL Server Storage Best Practice, High Availability Design, and SQL Server Maintenance Scripts sections.
3.5	January 2022	Updated the Best Practices Checklist section.
3.6	February 2022	Updated product naming.
3.7	June 2022	Updated the NUMA Architecture and Best Practices, CPU Hot-Add, Memory Reservations, and Drive Layout and Configuration sections and added the AHV Memory Hot-Plug section.
3.8	February 2023	Added the SQL Server VM Allocation section.
3.9	March 2023	Updated the CPU Hot-Add section and added the Microsoft SQL Server Filegroups section.

Version Number	Published	Notes
4.0	April 2023	Updated the Lock Pages in Memory and Large Pages section.

Nutanix Database Service

Nutanix Database Service (NDB) simplifies database management across hybrid multicloud environments for database engines like PostgreSQL, MySQL, Microsoft SQL Server, and Oracle Database, with powerful automation for provisioning, scaling, patching, protection, and cloning database instances. NDB helps customers deliver database as a service (DBaaS) and provide their developers an easy-to-use self-service database experience for on-premises and public cloud for both new and existing databases.

3. Understanding Microsoft SQL Server

SQL Server Overview

Microsoft SQL Server is a relational database management system (RDBMS) that supports a wide variety of business intelligence, analytics applications, and transaction processing in corporate IT environments. SQL Server is one of the most widely used database platforms in the world, with organizations typically deploying from dozens to hundreds of instances across the enterprise.

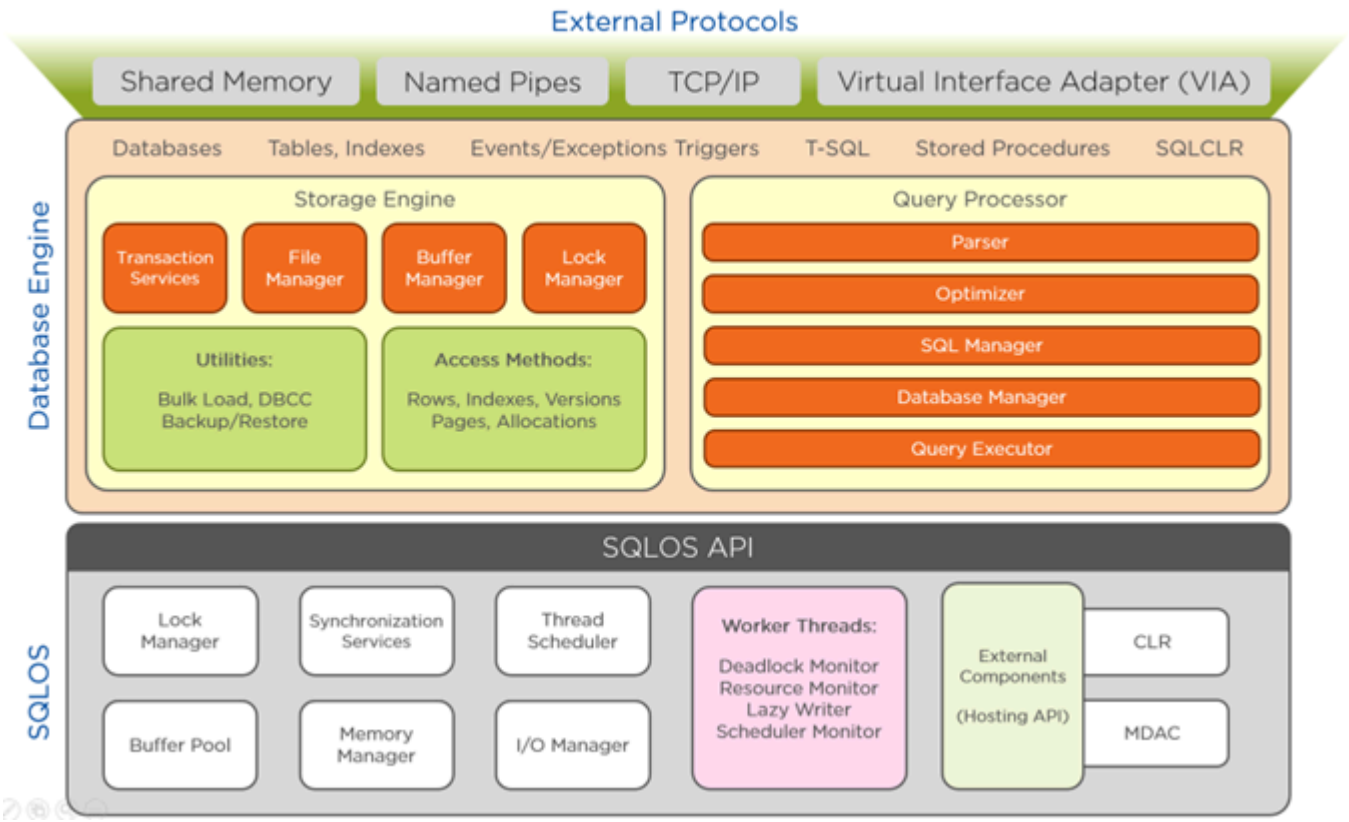

Figure 1: Microsoft SQL Server Architecture

The core component of Microsoft SQL Server is the SQL Server Database Engine, which controls data storage, processing, and security. It includes

a relational engine that processes commands and queries and a storage engine that manages database files, tables, pages, indexes, data buffers, and transactions. The Database Engine also creates and manages stored procedures, triggers, views, and other database objects.

The SQL Server Operating System (SQLOS) that underlies the Database Engine handles lower-level functions such as memory and I/O management, job scheduling, and locking data to avoid conflicting updates. A network interface layer sits above the Database Engine and uses Microsoft's Tabular Data Stream protocol to facilitate request and response interactions with database servers. At the user level, SQL Server DBAs and developers write T-SQL statements to build and modify database structures, manipulate data, implement security protections, and back up databases, among other tasks.

SQL Server Workload Types

When designing a new SQL Server database solution, it's important to understand the different SQL Server workload types. Each workload type has its own distinct characteristics and requirements that feed into how the overall solution is delivered.

Online Transaction Processing (OLTP)

OLTP is a data-modeling approach used to facilitate and manage typical business applications. Most applications are OLTP-based. Pure OLTP applications tend to have large numbers of small queries and usually a 70:30 split between reads and writes. Sustained CPU usage during work hours and high sensitivity to latency are common characteristics for OLTP workloads. Processors with faster base clock speeds and faster turbo speeds tend to perform better with OLTP queries. These workloads have more I/O operations per second than an equivalent data warehouse. With a single OLTP database, administrators see mostly sequential write activity to the transaction log file and more random write activity to the data files.

Online Analytic Processing (OLAP)

OLAP is an approach to answering multidimensional queries that was developed for Management Information Systems (MIS) and Decision Support

Systems (DSS). OLAP workloads have several different components, and the common workflow reads data from multiple sources, aggregates this data, processes the data, then returns it to customers, quite often in the shape of a report or dashboard. OLAP workloads are characterized by their use of a large sequential I/O profile, which means these workloads are geared toward throughput rather than latency. Processors with higher physical core counts are very useful for OLAP workloads, as is having a large amount of memory to process larger datasets. OLAP workloads tend to have a lot of random I/O, so flash-based storage for the cube files can be very beneficial.

Decision Support Systems (DSS)

Decision Support Systems are often used in conjunction with OLAP to access and manipulate data and support information-driven decision making in an organization. DSS workloads tend to have a few longer-running queries that are resource-intensive (CPU, memory, I/O) and generally occur during month-, quarter-, or year-end. DSS queries favor read over write, so it's important for the Nutanix platform hosting this workload type to have nodes sized to allow 100 percent of the dataset to reside in local flash. With Nutanix data locality, this sizing ensures that the system can provide that data as quickly as possible. This advantage places a premium on having processors with higher physical core counts.

Batch or Extract, Transform, and Load (ETL)

These workload types tend to be write-intensive, run during off-peak hours, tolerate contention better than OLTP workloads, and sometimes consume additional bandwidth. Organizations often run batch workloads at the end of the business day to generate reports about transactions that occurred during that day. We can break batch or ETL workloads down into three distinct phases:

1. Extract: The system contacts multiple sources and obtains data from these sources.
2. Transform: The system performs an action (or actions) on the obtained data to prepare it for loading into a target system.
3. Load: The data loads into the target system (often a data warehouse).

SQL Server Licensing

The customer is responsible for ensuring compliance with Microsoft licensing requirements when deploying SQL Server in a virtualized environment. There are some differences between licensing models depending on the SQL Server edition in use. It's important to deploy and maintain every solution in a supported fashion, so be sure that your specific combination of Nutanix AOS version, hypervisor type and version, and guest OS is supported. Consult the appropriate licensing guide for your supported SQL Server edition.

4. Physical Hardware Design

For organizations to successfully deploy SQL Server solutions, they must start with a solid foundation for the database and applications. We cover several considerations related to the platform's physical hardware in this section.

When you choose Nutanix, one of the key benefits is that the physical hardware (a node in Nutanix terminology) either arrives preconfigured from the factory (NX and OEM appliances) or can be automatically configured using the Nutanix Foundation process. When using third-party hardware from the Nutanix Hardware Compatibility Lists page, you may need to validate these physical hardware configurations and settings.

NUMA Architecture and Best Practices

Nonuniform memory access (NUMA) is a computer memory design where the memory access time depends on the memory location relative to the processor. With NUMA, a processor can access its own local memory faster than memory local to another processor or memory shared between processors. If a NUMA node wants to access nonlocal or remote memory, access times can be many times slower than local memory access. If the number of cores and the memory allocated to a SQL Server VM fits within a processor and its local memory, the hypervisor does its best to keep all memory access local.

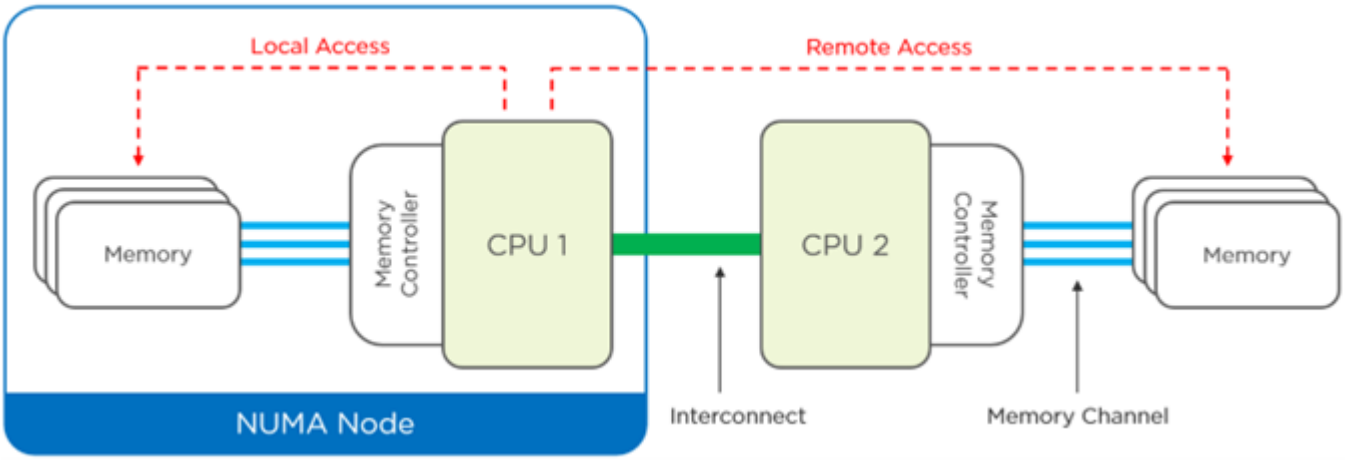

Figure 2: Logical View of NUMA Topology

VMware ESXi, Microsoft Hyper-V, and Nutanix AHV are all NUMA compatible, so they can present a virtual NUMA (or vNUMA) topology to the VM for NUMA-aware applications such as SQL Server to consume. For SQL Server VMs on AHV that don't fit within a NUMA boundary (number of cores + memory), enable vNUMA to give SQL Server visibility. For more details, see the section Enabling vNUMA on Virtual Machines in the AHV Administration Guide.

Understanding RAM Configuration

The physical memory of a server is critical in any virtualized environment, but particularly so for SQL Server, which requires a careful balance between memory capacity and memory speed. The number of memory channels available to a system depends on the generation of its CPUs. As an example, Intel Xeon processors with the Skylake microarchitecture have six memory channels per CPU, each with up to two DIMMs per channel (DPC), supporting a maximum of 12 DIMMs per CPU or 24 DIMMs per server (for a two-socket server). A previous generation (in Nutanix nodes, the Intel E5 Broadwell processor family) has four memory channels per CPU, each with three DPC, supporting a maximum of 12 DIMMs per CPU.

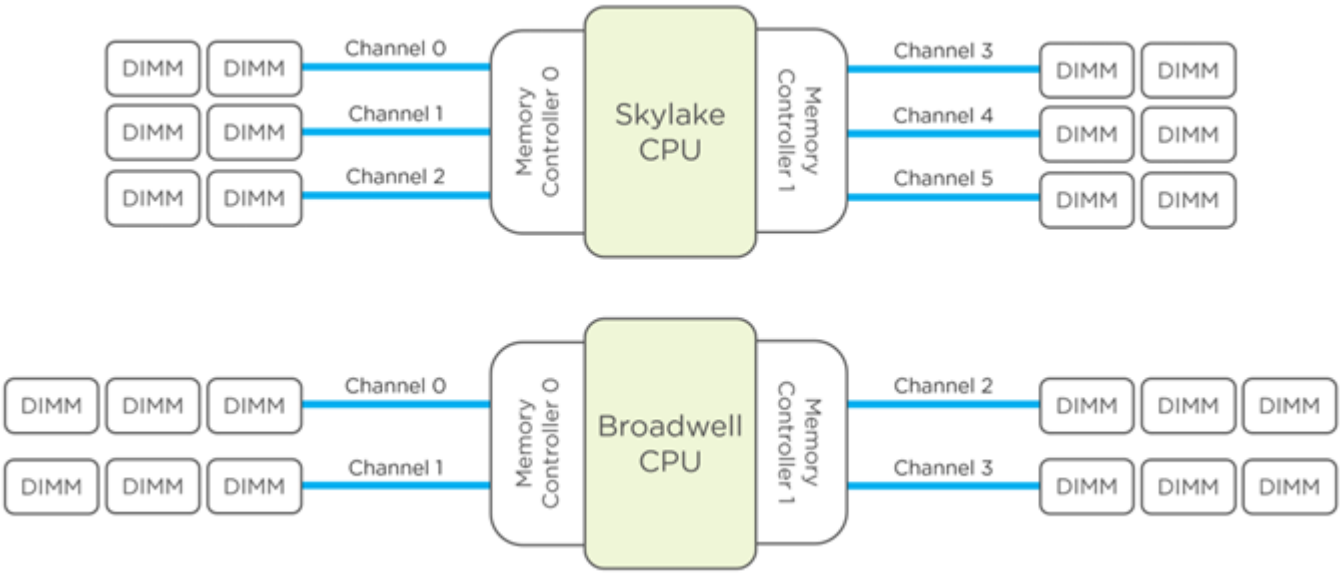

Figure 3: Intel Memory Architecture

To reduce stress on the memory controllers and CPUs, ensure that memory is in a Balanced configuration—where DIMMs across all memory channels and CPUs are populated identically. Balanced memory configurations enable optimal interleaving across all memory channels, maximizing memory bandwidth. To further optimize bandwidth, configure both memory controllers on the same physical processor socket to match. For the best system-level memory performance, ensure that each physical processor socket has the same physical memory capacity.

If DIMMS with different memory capacities populate the memory channels attached to a memory controller or if different numbers of DIMMs with identical capacity populate the memory channels, the memory controller must create multiple interleaved sets. Managing multiple interleaved sets creates overhead for the memory controller, which in turn reduces memory bandwidth.

For example, the following image shows the relative bandwidth difference between Skylake and Broadwell systems, each configured with six DIMMs per CPU.

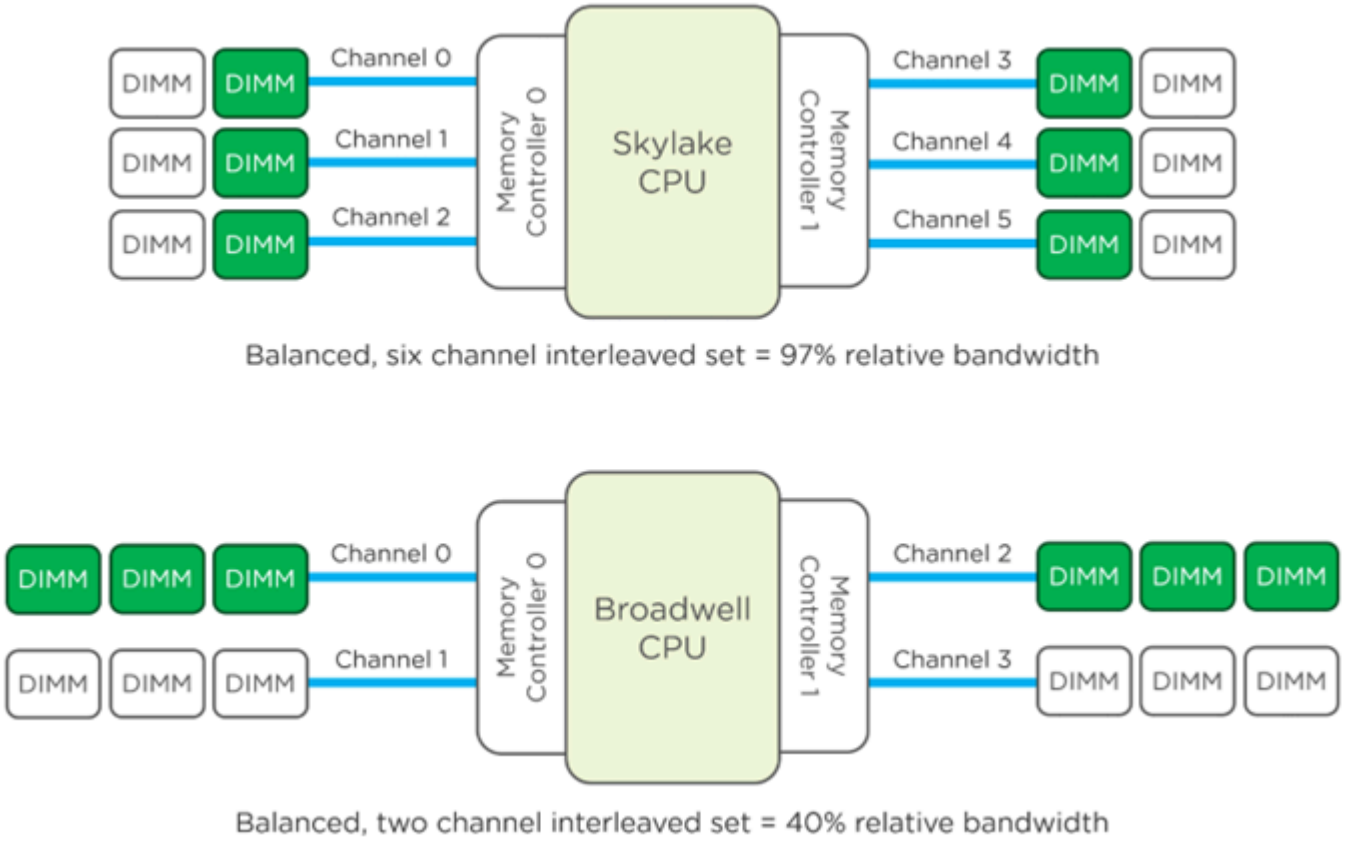

Figure 4: Example DIMM Population Between Skylake and Broadwell

Populating DIMMs across all channels delivers an optimized memory configuration. When following this approach with Skylake CPUs, populate memory slots in sets of three per memory controller. If you're populating a server that has two CPUs with 32 GB DIMMs, the six DIMMs across all channels per CPU deliver a total host capacity of 384 GB, providing the highest level of performance for that server. For a Broadwell system, a more optimized configuration has either four or eight DIMMs (as opposed to the six in Skylake). If you're populating a two-socket system with the same 32 GB DIMMs, eight DIMMs across all channels per CPU deliver a total host capacity of 512 GB, which is a higher performance configuration.

Note: Each generation of CPUs may change the way it uses memory, so it's important to always check the documentation from the hardware vendor to validate the configuration of the components.

BIOS or UEFI Firmware

To ensure that all components operate optimally and in a supported manner, Nutanix recommends always keeping system firmware up to date. Nutanix Life Cycle Manager (LCM) can help administrators keep firmware up to date across supported Nutanix and OEM platforms.

The Nutanix node installation process applies appropriate BIOS or UEFI settings. Third-party hardware platforms may need some additional configuration checks. As a best practice for Nutanix and SQL Server, ensure that NUMA and hyperthreading are enabled, along with CPU features such as Intel VMX and VT-x. Validate that all CPU C-states are disabled, as these power management policies can impact CPU efficiency and memory latency, resulting in poor application and VM performance.

Power Policy Settings

In addition to disabling the C-states at the BIOS or UEFI level, Nutanix recommends setting the host power management policy to High Performance for critical applications. This setting only applies to VMware ESXi and Microsoft Hyper-V; Nutanix AHV enables the high-performance mode automatically.

In ESXi, select the host and navigate to the Configure tab. Select Power Management and click High Performance.

In Hyper-V, open the Windows Control Panel and navigate to Power Options, then select High Performance. You can apply this setting globally using Group Policy.

Nutanix CVM Considerations

Every host in a Nutanix cluster has a CVM that consumes some of the host's CPU and memory to provide all the Nutanix services. The CVM can't live-migrate to other hosts, as the physical drives pass through to the CVM using the host hypervisor's PCI passthrough capability.

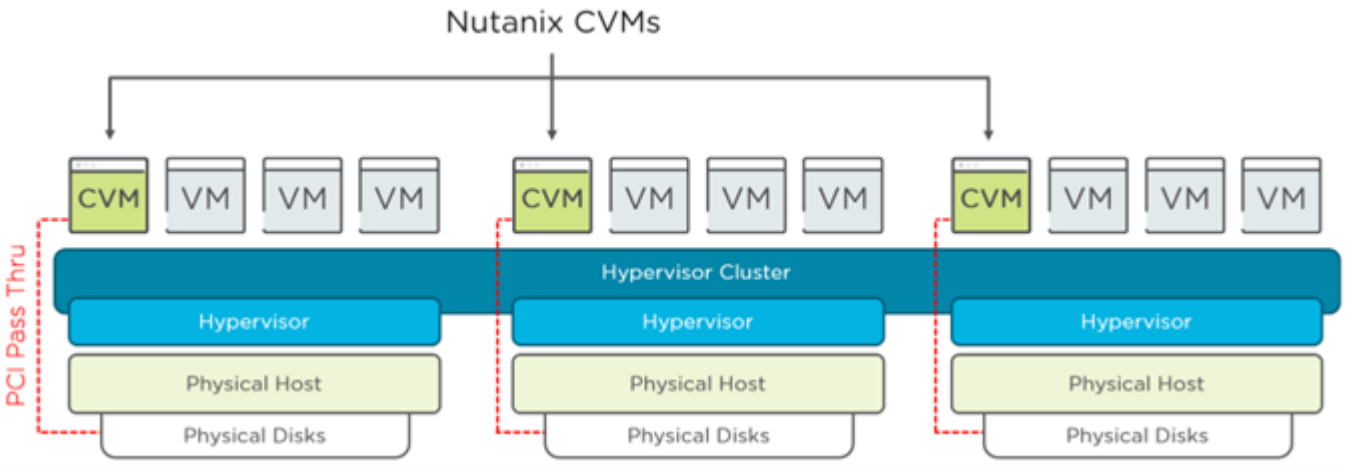

Figure 5: Nutanix CVMs in a Cluster

The size of the CVM varies depending on the deployment; however, the typical size is 8 vCPU and 32 GB of memory. During initial cluster deployment, Nutanix Foundation automatically configures the vCPU and memory assigned to the CVM. Consult your Nutanix partner and Nutanix Support if you need to make any changes to these values to ensure that the configuration is supported.

vCPUs are assigned to the CVM and not necessarily consumed. If the CVM has a low-end workload, it uses the CPU cycles it needs. If the workload becomes more demanding, the CVM uses more CPU cycles. Think of the number of CPUs you assign as the maximum number of CPUs the CVM can use, not the number of CPUs immediately consumed.

When you size any Nutanix cluster, ensure that there are enough resources in the cluster (especially failover capacity) to handle situations where a node fails or goes down for maintenance. Sufficient resources are even more important when you're sizing a solution to host a critical workload like Microsoft SQL Server.

From Nutanix AOS 5.11 onward, AHV clusters can include a new type of node that doesn't run a local CVM. These compute-only (CO) nodes are for very specific use cases, so engage with your Nutanix partner or Nutanix account manager to see if CO is advantageous for your workload. Because standard nodes (running CVMs) provide storage from AOS storage, CO nodes can't benefit from data locality.

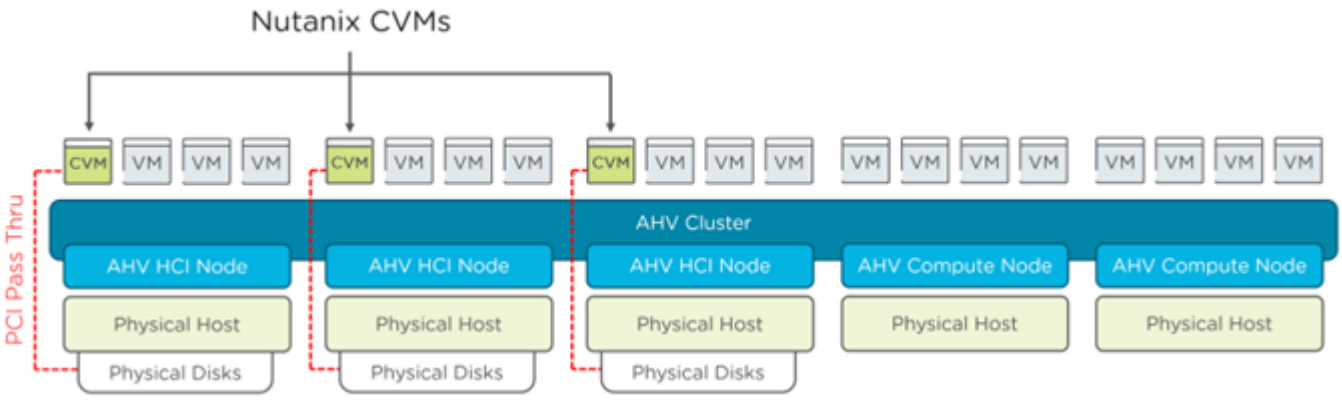

Figure 6: Nutanix AHV Cluster with HCI and Compute Nodes

Note: CO nodes are only available when running Nutanix AHV as the hypervisor.

5. Virtual Hardware Design

CPU Definitions

In a virtualized environment, many different components make up a CPU for an application such as SQL Server to consume. The following diagram shows the physical and logical constructs of a CPU.

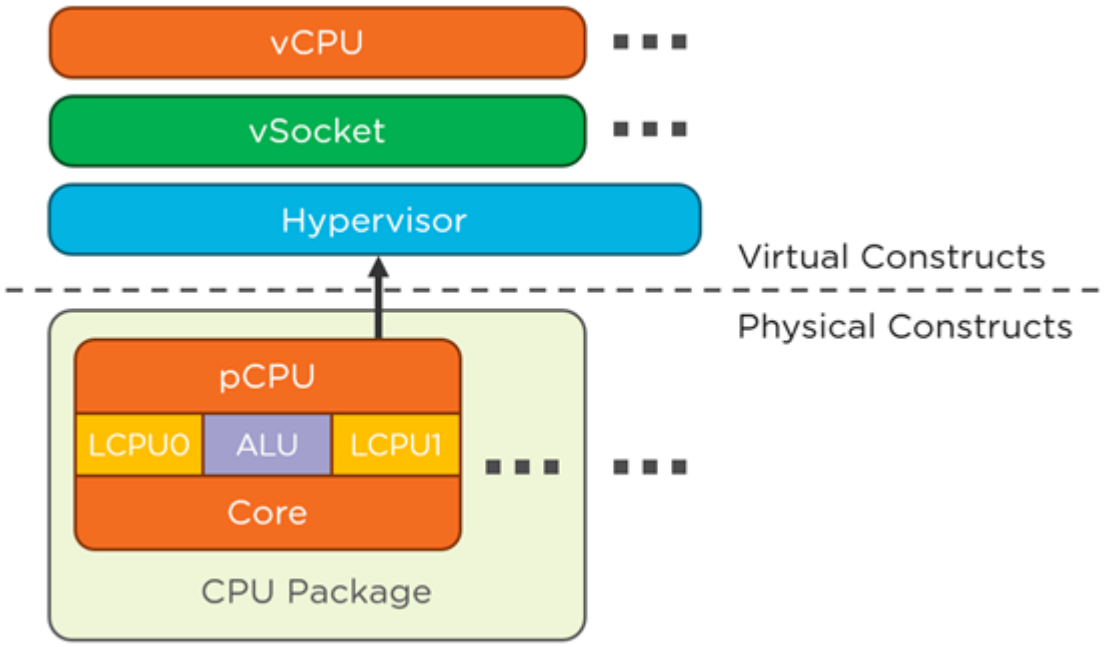

Figure 7: CPU Terminology

CPU package

The CPU package is the physical device that contains all the components and cores and sits on the motherboard. Many different types of CPU packages are available, ranging from low-core, high frequency models to high-core, low-frequency models. The right balance between core and frequency depends on the requirements of the SQL Server workload, considering factors such as the type of workload, license, and query

(single-threaded or multithreaded). There are typically two or more CPU packages in a physical host.

Core

A CPU package typically has two or more cores. A core is a physical subsection of a processing chip and contains one interface for memory and peripherals.

Logical CPU (LCPU)

The LCPU construct presented to the host operating system is a duplication of the core that allows it to run multiple threads concurrently using hyperthreading (covered in the next section). LCPUs don't have characteristics or performance identical to the physical core, so don't consider an LCPU a core in its own right. For example, if a CPU package has 10 cores, enabling hyperthreading presents 20 LCPUs to the guest OS.

Arithmetic logic unit (ALU)

The ALU is the heart of the CPU, responsible for performing mathematical operations on binary numbers. Typically, the multiple local CPUs on the core share one ALU.

Hypervisor

In a virtualized environment, the hypervisor is a piece of software that sits between the hardware and one or more guest operating systems. Each of these guest operating systems can run its own programs, as the hypervisor presents to it the host hardware's processor, memory, and resources.

vSocket

Each VM running on the hypervisor has a vSocket construct. vCPUs virtually plug in to a vSocket, which helps present the vNUMA topology to the guest OS. There can be many vSockets per guest VM or just one.

vCPU

Guest VMs on the hypervisor use vCPUs as the processing unit and map them to available LCPUs through the hypervisor's CPU scheduler. When determining how many vCPUs to assign to a SQL Server, always size assuming 1 vCPU = 1 physical core. For example, if the physical host

contains a single 10-core CPU package, don't assign more than 10 vCPU to the SQL Server. When considering how much to oversubscribe vCPUs in the virtual environment, use the ratio of vCPU to cores, not LCPUs.

Hyperthreading Technology

Hyperthreading is Intel's proprietary simultaneous multithreading (SMT) implementation used to improve computing parallelization. Hyperthreading's main function is to deliver two distinct pipelines into a single processor implementation core. From the perspective of the hypervisor running on the physical host, enabling hyperthreading makes twice as many processors available.

In theory, because we ideally want to use all available processor cycles, we should use hyperthreading all the time. However, in practice, using this technology requires some additional consideration. There are two key issues when assuming that hyperthreading simply doubles the amount of CPU on a system:

1. How the hypervisor schedules threads to the pCPU.
2. The length of time a unit of work stays on a processor without interruption.

Because a processor is still a single entity, appropriately scheduling processes from guest VMs is critical. For example, the system shouldn't schedule two threads from the same process on the same physical core. In such a case, each thread takes turns stopping the other to change the core's architectural state, with a negative impact on performance. SQL Server complicates this constraint because it schedules its own threads; depending on the combination of SQL Server and Windows OS versions in use, SQL Server may not be aware of the distinction between physical cores or know how to handle this distinction properly. There is also a hypervisor CPU scheduling function to keep in mind, which is abstracted from the guest OS altogether.

Because of this complexity, you must size for physical cores and not hyperthreaded cores when sizing SQL Server in a virtual environment. For mission-critical deployments, start with no vCPU oversubscription. SQL Server deployments—especially OLTP workloads—benefit from this approach.

CPU Hot-Add

The effect of hot-adding vCPUs is particularly relevant to database VMs, which can be NUMA-wide. Because we generally size databases using vCPUs capable of handling peak workloads with an additional buffer, hot-add might not be an urgent use case. However, if you need to hot-add vCPUs beyond a NUMA boundary, what you lose in vNUMA benefits depends on the workload and on NUMA optimization algorithms specific to the database vendor and the version of VMware vSphere you're using. To determine whether the performance tradeoff is warranted in your specific circumstances, VMware recommends determining the NUMA optimization benefits based on your own workload before setting the hot-add vCPU function.

Consider NUMA boundaries before hot-adding vCPUs. Hot-adding vCPUs can disable vNUMA in VMware vSphere. For more information, see VMware KB 2040375.

CPU Ready Time

One of the key statistics that administrators use to determine overall health in a virtualized environment is CPU ready time. This metric tracks the amount of time a VM is ready to process a thread but is instead waiting for the hypervisor scheduler to provide access to a physical processor. When the guest VM (and thus SQL Server) is in this waiting state, no operations occur and the database's overall processing throughput drops.

Host vCPU oversubscription is the main reason administrators might see a high CPU ready time—if there is a high ratio of vCPUs to available physical cores and the guest VMs need to use CPU, then the hypervisor scheduler is under pressure to allot time to the guest's vCPU. The only effective way to avoid this contention is to right-size SQL Server VMs to use what they need, not some larger amount.

As a rule, if CPU ready time is over 5 percent for a given VM, there is cause for concern. Always ensure that the size of the physical CPU and the guest VM is in line with SQL Server requirements to avoid this problem.

Memory Configuration

Virtual Memory Overview

In addition to providing CPU time for guest VMs, the hypervisor also virtualizes VM memory to isolate workloads from each other and to provide a contiguous, zero-based memory space for each guest operating system. Each supported hypervisor uses different techniques to manage memory but the overall concept is the same.

The following diagram demonstrates a conceptual overview of memory management in a virtual environment.

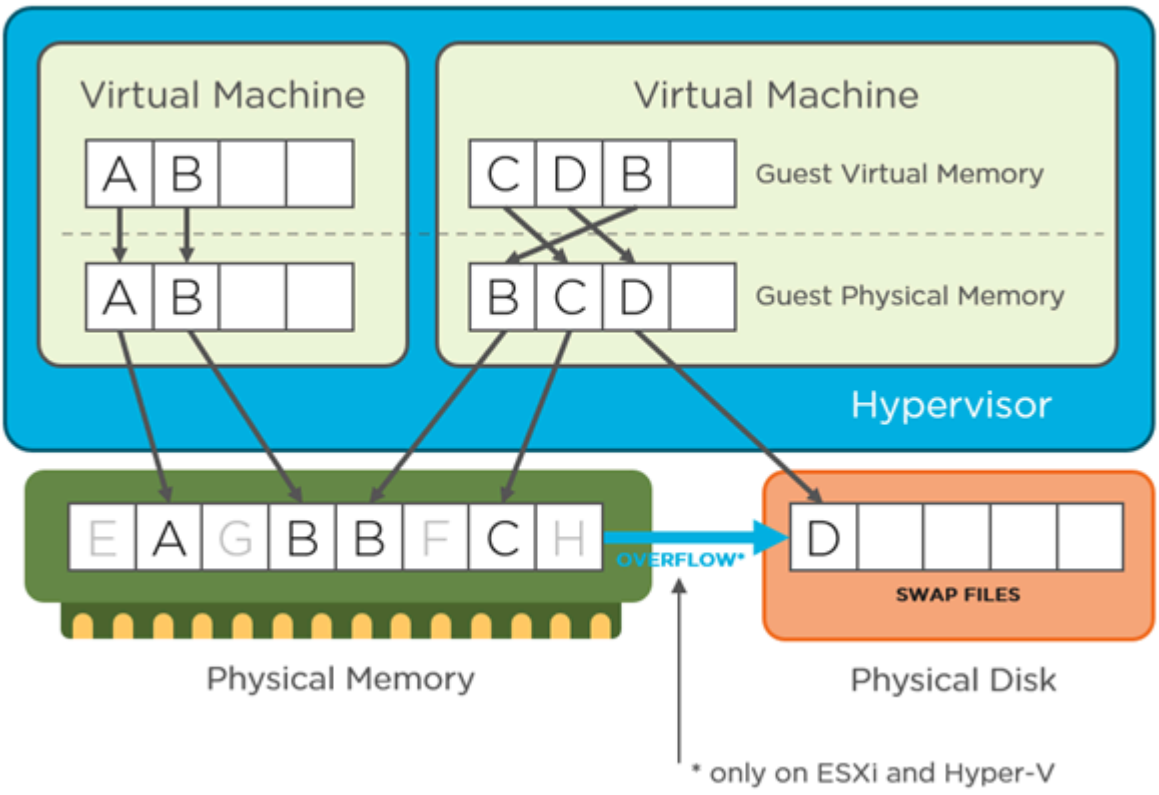

Figure 8: Virtual Memory Overview

One of the key differences between Nutanix AHV and VMware ESXi or Microsoft Hyper-V is that AHV doesn't support swapping VM memory to disk. With ESXi, you can assign more memory to all VMs on a host than there is physical host

memory. This overcommitment can be dangerous, as physical RAM is much faster than even the fastest storage available on Nutanix. Similarly, with Hyper-V, technologies such as Dynamic Memory and Smart Paging allow the system to share memory allocation between all guest VMs with physical disks as an overflow.

Memory Reservations

Because memory is the most important resource for SQL Server, don't oversubscribe memory for this application (or for any business-critical applications). As mentioned earlier, Nutanix AHV supports memory overcommit, so all memory is effectively reserved for all VMs.

When you enable memory overcommit, AHV calculates an appropriate memory size for each VM based on its memory usage. Memory overcommit enables the host to automatically detect memory underutilization on a VM and combine any excess or underutilized memory, which the host can then use to provision memory for another VM or to create another VM. With this feature, you can achieve virtualized optimization of host memory. Memory overcommit is particularly useful in test and development environments to verify VM performance.

When using VMware ESXi or Microsoft Hyper-V, ensure that the host memory is always greater than the sum of all VMs. If there is memory contention on the hypervisor, guest VMs could start to swap memory space to disk, which has a significant negative impact on SQL Server performance.

If you're deploying SQL Server in an environment where oversubscription may occur, Nutanix recommends reserving 100 percent of all SQL Server VM memory to ensure a consistent level of performance. This reservation prevents the SQL Server VMs from swapping virtual memory to disk; however, other VMs that don't have this reservation may swap.

Memory Hot-Plug and Dynamic Memory

Nutanix AHV, VMware ESXi, and Microsoft Hyper-V offer built-in memory management techniques that allow you to add memory to a running VM without needing to reboot. AHV and VMware have memory hot-plug and Microsoft has Dynamic Memory.

AHV Memory Hot-Plug

Memory is hot-pluggable on guest VMs running on AHV, which means that you can increase the memory allocation on your VMs while they're turned on. To do so, you can use Prism Element (see the Managing a VM (AHV) section of the Prism Web Console Guide) or Prism Central (see the Managing a VM (AHV) and Managing a VM (Self Service) sections of the Prism Central Guide) or you can use a simple aCLi command:

```
$ nutanix@cvm$ acli vm.update vm-name memory=new_memory_size
```

Replace vm-name with the name of the VM and new_memory_size with the memory size.

ESXi Memory Hot-Plug

With vSphere 6.5 and later, enabling memory hot-plug (which isn't a default setting) adds memory evenly to all vNUMA nodes, preserving the benefits of vNUMA topology for SQL Server.

It can be beneficial to enable this feature for SQL Server VMs, as you can add more memory to the SQL Server VMs without restarting them; however, you do need to restart the SQL Server instance (service). Pay close attention to the total VM memory allocation, as it's best to fit a VM within the memory footprint of one NUMA node. This arrangement provides optimal memory performance across the entire address space. Also, be aware of the total memory allocated to all VMs so you don't oversubscribe memory on the host and potentially start swapping to disk.

Once you have increased the memory on VMs where hot-plug is enabled, remember to adjust the SQL Server maximum memory setting (outlined in the section SQL Server Memory Configuration in this document). Adjusting this value doesn't require you to restart the SQL Server service.

Hyper-V Dynamic Memory

Instead of allocating a fixed amount of memory to each VM with the potential for overflow to disk once physical host memory is exhausted, Dynamic Memory allows the system to see RAM as a shared resource that it can dynamically and automatically divide between the running VMs. Hyper-V doesn't overallocate memory on hosts the way VMware ESXi does. With Dynamic Memory enabled,

Hyper-V can dynamically adjust VM memory in response to changes in the workload of the running VMs. This capability allows the system to distribute memory resources more efficiently between active VMs.

As with VMware's CPU hot-add, enabling Dynamic Memory in the settings of a VM disables vNUMA. If memory is split between NUMA nodes, there can be up to a 30 percent drop in performance. Although enterprise editions of SQL Server can dynamically add or remove memory, failing to maintain the vNUMA topology can lead to inconsistent performance. For this reason, Nutanix recommends against using Dynamic Memory for SQL Server workloads.

Network Configuration

Networking Overview

Along with compute resources such as CPU and memory, the network is also a critical piece of any successful SQL Server deployment. Without a robust, fully redundant network, access to databases and supporting applications can be slow, intermittent, or even unavailable.

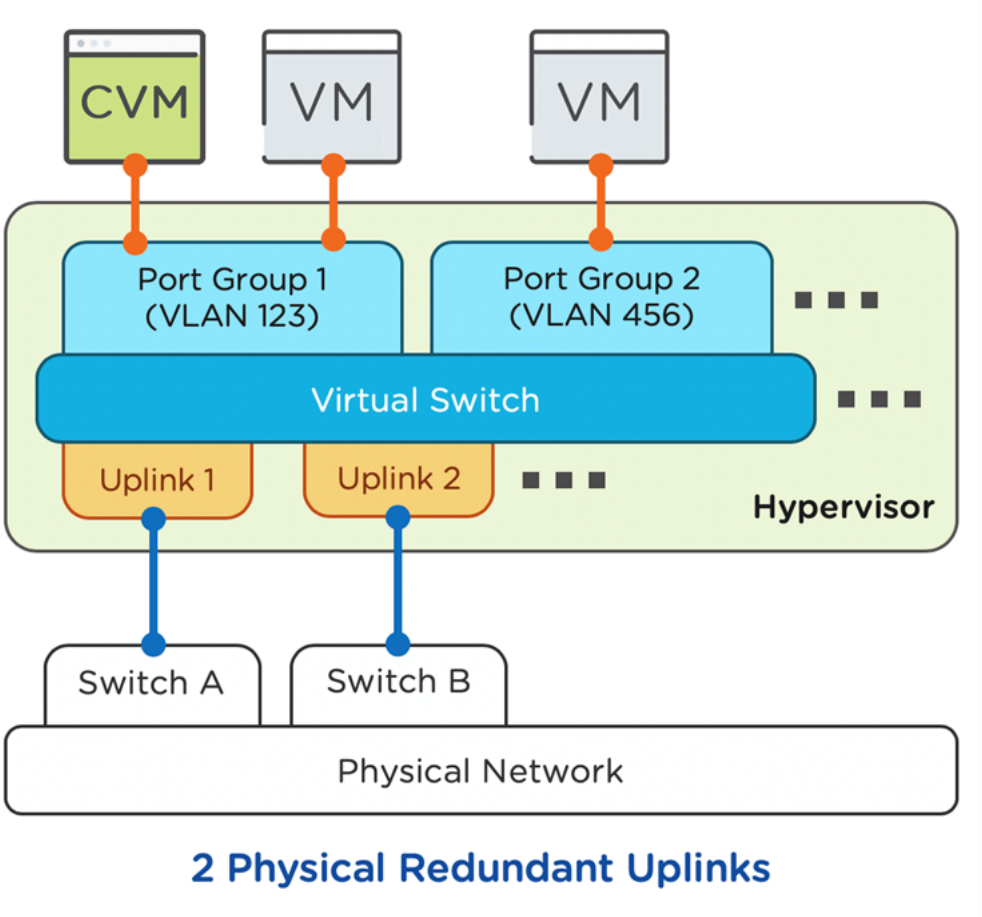

Figure 9: Virtual Networking Overview

Although each hypervisor implements virtual networking constructs slightly differently, the basic concepts remain the same.

Networking Best Practices

Virtual network design isn't specific to SQL Server. The following are generic best practices for designing virtual networks:

- Always use redundant physical network connections to mitigate any failure of either a physical switch or physical NIC interface on the host.
- Always size network bandwidth to ensure that the network can maintain the same level of bandwidth during a component failure. For example, if the physical servers have two 10 GB interfaces, don't configure teaming policies

to provide 20 GB of bandwidth to the VMs. If the VMs expect 20 GBps of bandwidth and one of the links fails, performance then drops by 50 percent.

- Consider using hypervisor-based network I/O control to ensure that transient high-throughput operations such as vMotion or live migration don't negatively impact critical traffic.
- For VMware-based environments, consider using the VMware Distributed Switch. This construct provides many benefits in a VMware environment, including the ability to use technologies such as LACP and maintain TCP state during vMotion.

For more detailed information about networking in a Nutanix environment for supported hypervisors, refer to the following resources:

- Nutanix AHV Networking best practice guide
- vSphere Networking with Nutanix best practice guide
- Hyper-V 2016 Networking with Nutanix best practice guide

Use of Jumbo Frames

The Nutanix CVM uses the standard Ethernet MTU (maximum transmission unit) of 1,500 bytes for all the network interfaces by default. The standard 1,500 byte MTU delivers excellent performance and stability. Nutanix doesn't support configuring the MTU on a CVM's network interfaces to higher values.

You can enable jumbo frames (MTU of 9,000 bytes) on the physical network interfaces of AHV, ESXi, or Hyper-V hosts and user VMs if the applications on your user VMs require them. If you choose to use jumbo frames on hypervisor hosts, enable them end to end in the desired network and consider both the physical and virtual network infrastructure impacted by the change.

Storage Configuration

Nutanix Volumes

Nutanix Volumes is a native scale-out block storage solution that provides direct block-level access via the iSCSI protocol to AOS storage. It enables

enterprise applications running on external servers to benefit from the hyperconverged Nutanix architecture. Nutanix Volumes also allows guest VMs on any hypervisor to support clustered file systems.

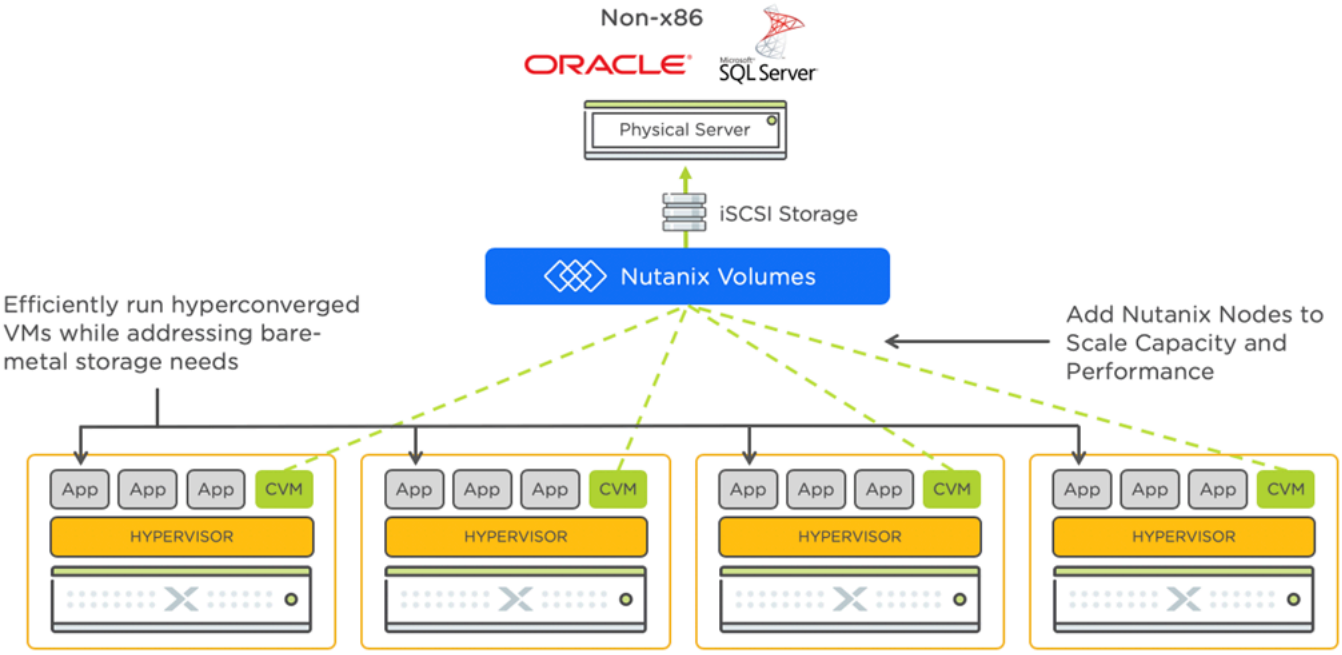

Figure 10: Nutanix Volumes

Nutanix Volumes supports SCSI-3 persistent reservations for the shared storage used by SQL Server failover cluster instances (FCIs). Nutanix manages storage allocation and assignment for Volumes through a construct called a volume group (VG). A VG is a collection of virtual disks (vDisks) presented as available block devices using the iSCSI protocol. All Nutanix CVMs in a cluster participate in presenting these VGs, creating a highly redundant and scalable block storage solution. When you create a Windows failover cluster for SQL Server, you need Nutanix Volumes because Windows failover clusters require shared storage, but when you deploy SQL Always On availability groups (AGs), you don't need Volumes because AGs don't require shared storage.

Nutanix Volumes seamlessly manages failure events and automatically load balances iSCSI clients to take advantage of all cluster resources. iSCSI redirection controls target path management for vDisk load balancing and path resiliency. Instead of configuring host iSCSI client sessions to connect

directly to all CVMs in the Nutanix cluster, Volumes uses the Data Services IP address. This design allows administrators to add Nutanix nodes to the cluster in a nondisruptive manner without needing to update every iSCSI initiator.

For more details on Nutanix Volumes, refer to the Nutanix Volumes best practice guide.

Nutanix Volume Group Load Balancer (AHV Only)

Nutanix AHV makes the consumption of VGs easier than other hypervisors or physical servers. VGs can connect directly to a VM running on AHV without requiring you to configure in-guest iSCSI initiators.

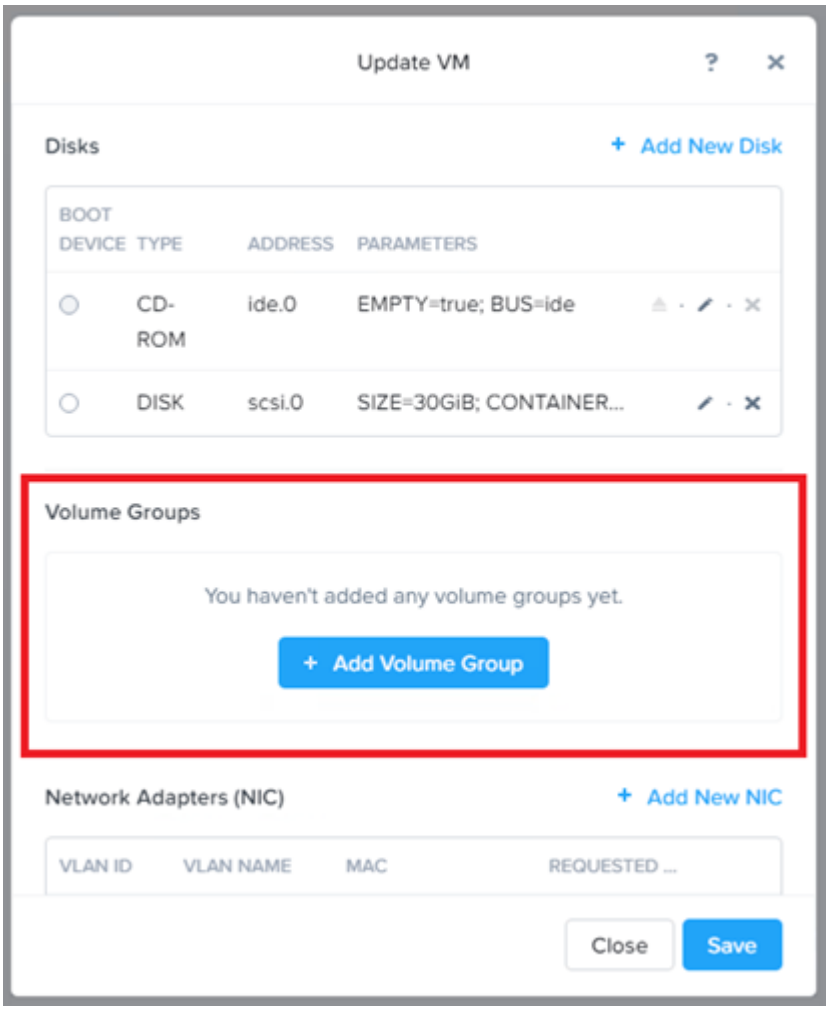

Figure 11: Adding a Volume Group in Nutanix AHV

A VG directly connected to an AHV VM exhibits the same behavior as a VG presented via iSCSI, with one exception: with VGs, vDisks aren't load balanced across all CVMs in the cluster by default. This lack of load balancing is equivalent to normal VMs running on an AHV host and isn't a concern for most workloads. Nutanix recommends starting with the Volume Groups with Load Balancing (VGLB) feature disabled. Nutanix data locality and a correctly sized SSD tier provide optimal performance for most SQL Server workloads without enabling VGLB.

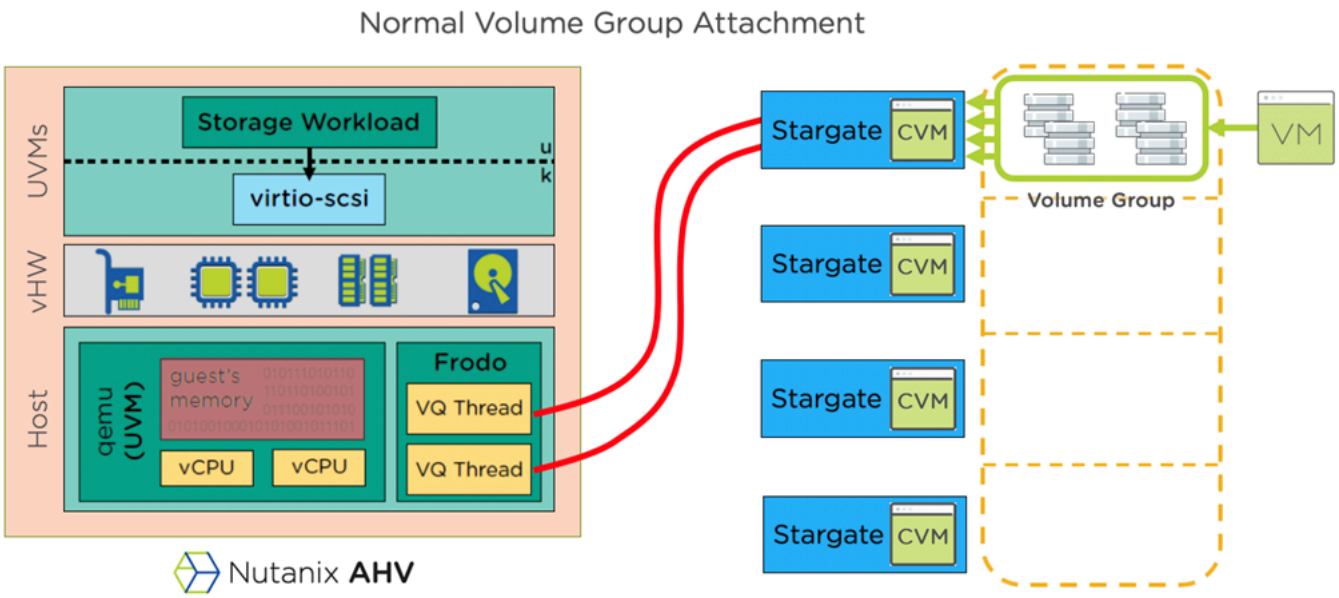

Figure 12: Nutanix AHV Volume Groups (Not Load Balanced)

There may be instances where VGLB can improve performance, especially with large workloads where the size of the dataset is greater than the size of the underlying host or where throughput is higher than what a single node can provide, as with OLAP and DSS workloads. Particularly when physical systems with a small number of drives are hosting VMs and volume groups with a high number of vDisks, the solution is to use VGLB. This feature isn't enabled with VGs by default, so you must enable it on a per-VM basis.

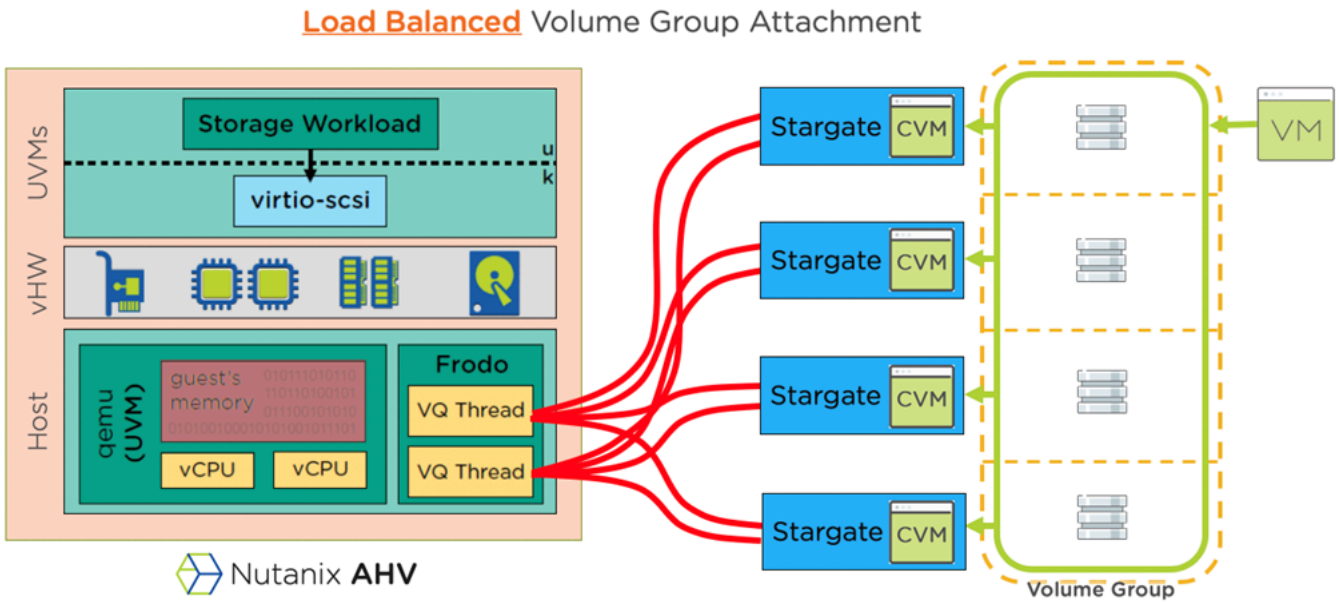

Figure 13: Nutanix AHV Volume Groups (Load Balanced)

As illustrated in the previous image, enabling VGLB removes the bottleneck of the local Nutanix CVM so that all CVMs can participate in presenting primary read and write requests. While this approach can provide great performance benefits overall, using VGLB adds a small amount of I/O latency because some data locality is lost when the vDisks shard across the CVMs on the cluster. In the right use case, this cost is generally worth it, as VGLB gives the system a far greater amount of available bandwidth and CPU resources to serve I/O.

SQL Server VM CPU Allocation

Depending on the edition of SQL Server deployed, Microsoft sets licensing limits on the number of CPUs that you can allocate to each SQL Server VM instance. Refer to Microsoft documentation for the latest information on scale limits for each SQL Server edition.

It's important to note how physical CPUs map to vCPUs in a hypervisor context. Refer to the Configuring vCPU for Microsoft SQL Server Standard Edition tech note for the considerations to keep in mind when deploying SQL Server Standard edition in a virtual environment.

SQL Server Storage Best Practices

The following sections outline the storage best practices for running SQL Server on Nutanix.

Drive Layout and Configuration Best Practices

Nutanix is a scale-out platform that distributes disks and I/O across a shared-nothing cluster that scales in a linear fashion. It's a general best practice to create many smaller disks as opposed to a few large disks. One key reason for this recommendation is that each vDisk on Nutanix (regardless of the hypervisor) contains a fixed amount of access to the random write buffer on the underlying host (called the oplog). The optimal number and size of the virtual disks and their corresponding SQL Server data or log files depends on the size and activity of the database. Adding more files and properly aligning them with virtual disks can increase the I/O performance.

In keeping with the scale-out mindset, it's also important to ensure that the different types of disks (system, data files, TempDB, and logs) are all placed across multiple virtual storage controllers. This consideration isn't relevant for Nutanix AHV, as AHV doesn't have the construct of a virtual storage controller. However, for VMware ESXi and Microsoft Hyper-V, ensure that there are multiple virtual SCSI adapters for SQL Server.

As a starting point for storage, Nutanix recommends the following:

- Separate drives for the OS and SQL Server binaries (and backup or restore drive, if used). Place these drives together on the same virtual storage controller (for example, controller 0).
- At least two drives for database data files, with each drive containing multiple data files. Place these drives together on the same virtual storage controller but ensure that it's a different virtual storage controller than the one containing the OS and binaries (for example, controller 1). Read more in the section titled SQL Server Data Files Best Practices.
- Two TempDB data drives, with one TempDB data file per vCPU across the drives, up to eight TempDB data files. Place these vDisks on their own virtual storage controller (for example, controller 2). Read more in the section titled TempDB Sizing and Placement Best Practices.

- One drive for database log files and another drive for the TempDB log file, with each drive on its own virtual storage controller (for example, controller 3). Read more in the section titled SQL Server Log Files Best Practices.
- Ensure that user databases and system databases are on separate drives.

The following diagram shows the suggested initial drive layout for SQL Server VMs on Nutanix.

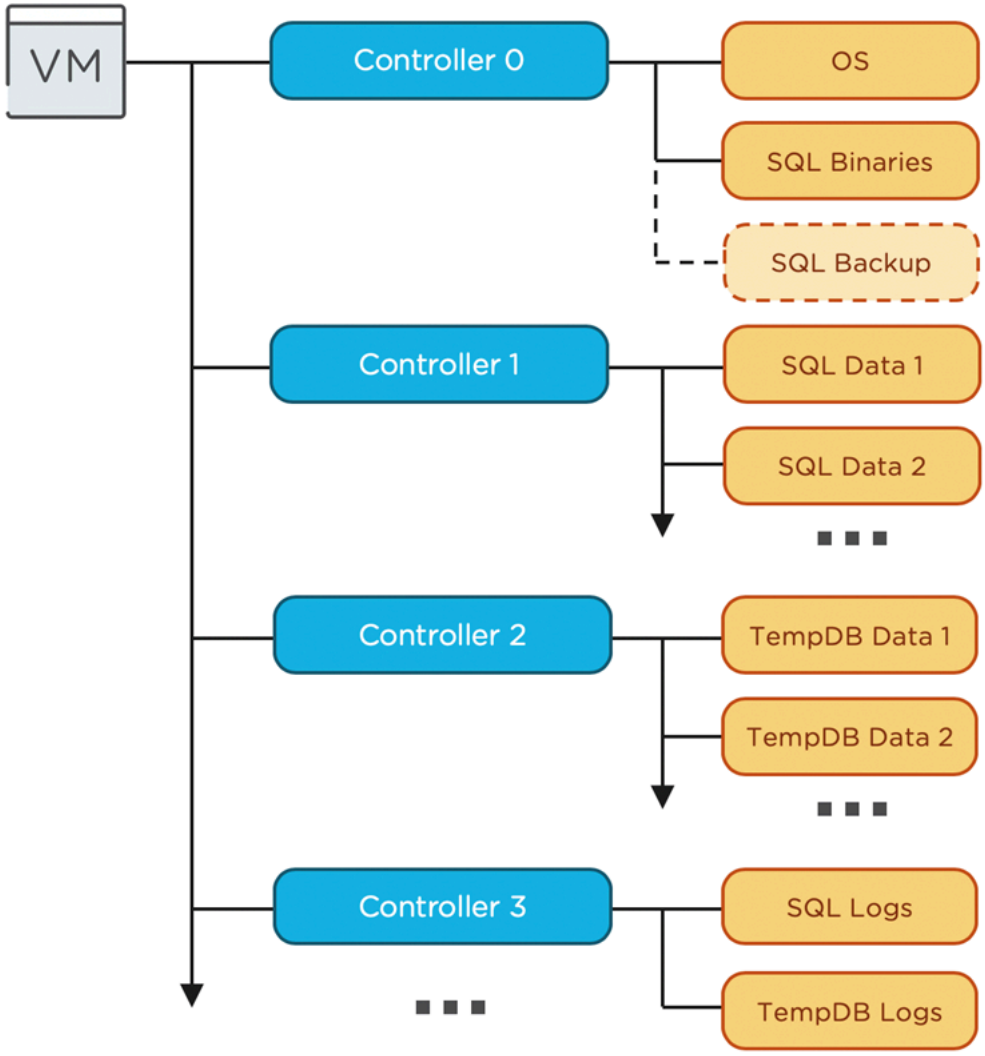

Figure 14: Initial Drive and Controller Layout

In addition to providing performance and scalability, maintaining this separation helps with manageability, as potential problems are easier to isolate. For example, when you separate TempDB onto its own disks, you can configure the files to grow and fill the disk without worrying about space requirements for other files (within certain limits—filling the disk to 100 percent capacity still causes issues for TempDB). The more separation you can build into the solution, the easier it is to correlate potential disk performance issues with specific database files.

In a VMware environment, Nutanix recommends using the VMware paravirtual (PVSCSI) controller for all data and log drives. For the OS drive, it's simpler to keep the default LSI SAS adapter; otherwise, the system must load the PVSCSI drivers to allow the OS to boot, which you can do when you create the Windows template.

In a Hyper-V environment, if you are using Generation 1 VMs, use SCSI disks for all drives except the OS. For Generation 2 VMs, use SCSI disks for all drives including the OS.

SQL Server Data Files Best Practices

To make the most out of a platform build, you need to employ a scale-out approach to the disk and data files layout. Our recommendation is aligned with what Microsoft suggests, as splitting the database across multiple files across multiple disks provides better performance compared with a configuration based on a single data file and single disk. For example, you could split a SQL Server with 4 vCPU hosting a 500 GB database into four 125 GB database files spread across two vDisks. In this configuration, it's important to grow files equally. For SQL Server 2016 and newer versions, use the following T-SQL command:

```
$ ALTER DATABASE (database name)
$ MODIFY FILEGROUP [PRIMARY] AUTOGROW_ALL_FILES
$ GO
```

For versions prior to SQL Server 2016, apply trace flag 1117 at SQL Server startup.

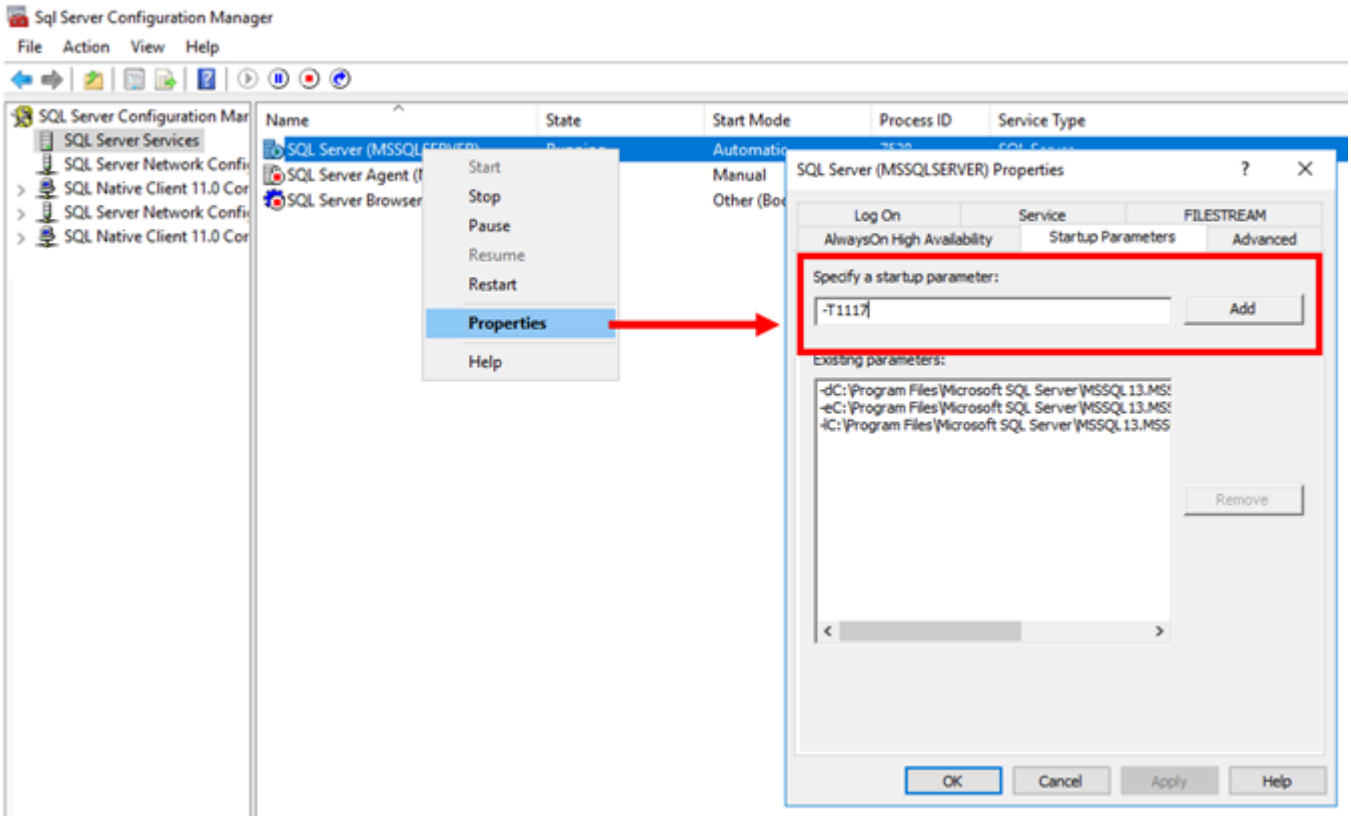

Figure 15: Applying SQL Server Startup Parameters

To avoid unnecessary complexity, add files to databases only if there is database page I/O latch contention. To look for contention, monitor the PAGEIOLATCH_XX values and spread the data across more files as needed. Several other factors, such as memory pressure, can also cause PAGEIOLATCH_XX latency, so investigate the situation thoroughly before adding files.

Unlike transaction log files, SQL Server accesses data files in parallel, and access patterns can be highly random. Spreading large databases across several vDisks can improve performance. It's best to create multiple data files before writing data to the database so that SQL Server's proportional fill algorithm spreads data evenly across all data files from the beginning.

You should only use volume managers (dynamic disks, storages spaces) with the OS as last resort after careful testing. Using multiple data files across

multiple disks is the recommended and default approach. Also, don't use autoshrink on database files, as the resulting fragmentation can reduce performance.

> Note: Nutanix recommendations and our tested configuration rely on using multiple vDisks to host a database's data files. Nutanix advises against using any in-guest OS volume manager. Using a volume manager may negatively impact performance or cause issues with third-party applications (for example, backup tools).

It isn't uncommon for SQL Server to host several small databases, none of which are particularly I/O intensive. In this case, the design could have only one or two data files per database. These files could easily fit on one or two disks and not spread over multiple disks. The goal of disk and file design is to keep the solution as simple as possible while meeting business and performance requirements.

SQL Server Log Files Best Practices

The total size reserved for the database transaction log file should be based on the high-water mark before the next backup and log truncation time. If the system performs daily backups, estimate how much log space the database is likely to require in 24 hours. The log file space requirement depends heavily on the application, so measure it appropriately for your specific environment.

If the databases have an unusually high number of activities that influence the size of the log file (inserts, updates, deletes, changes in the recovery model), it can be beneficial to perform regular backups more often than every 24 hours. Increasing backup frequency on Nutanix doesn't require a change in the existing backup strategy or design.

Physical transaction logs are made up of several smaller units called virtual log files (VLFs). VLFs play a critical part in performing database backups and recovery operations. The number of VLFs is determined at log creation and whenever there is any file growth. If the VLFs are too small, database recovery and other operations can be very slow. If the VLFs are too large, backing up and clearing logs can be slow.

Optimal VLF size is between 256 MB and 512 MB. Even if the logs reach 2 TB in size, they shouldn't contain more than 10,000 VLFs. To preserve this limit, preset the log file to either 4 GB or 8 GB and grow it by increments of the

same starting amount to reach the desired log size. If you need a 128 GB log file, initially create an 8 GB log file and grow it 15 times in 8 GB increments. Configure the autogrow value to the same base (4 GB or 8 GB) to maintain the same VLF size. In SQL Server Management Studio, issue the DBCC LOGINFO (dbName) command to return the number of VLFs in a database.

Remember that, unlike data files, SQL Server log files are written to in a sequential or "fill and spill" manner. Therefore, using multiple log files spread across several vDisks isn't advantageous and just makes the solution more complex than necessary. Because SQL Server only accesses one log file at any given time, there's no reason to use more than one log file.

TempDB Sizing and Placement Best Practices

One of the most important elements of maximizing SQL Server performance is optimizing TempDB. Applications can use TempDB as a scratch space and in most cases shouldn't rely on a single TempDB data file. If there are fewer than 8 vCPU, the recommendation is to configure the number of TempDB data files to be equal to the number of assigned vCPUs. If there are more than 8 vCPU, start with eight TempDB data files and look for contention for in-memory allocation (PAGELATCH_XX). Scale up the number of TempDB files in increments of four until the contention has been eliminated. During the installation, SQL Server 2016 automatically allocates TempDB files equal to the number of assigned vCPUs, up to 8 vCPU.

The size and autogrowth settings for TempDB data files are also important. SQL Server allocates space in TempDB data files proportionally based on their size, so it's important to create all TempDBs to be the same size. Size them properly up front to accommodate application requirements; proper sizing is generally between 1 percent and 10 percent of the database size. During a proof-of-concept deployment, monitor TempDB size and use the high-water mark as the starting point for the production sizing. If you need more TempDB space in the future, grow all data files equally. You can rely on autogrowth but autgrowth can grow the data files to a size beyond the free space available on the disk, so we don't recommend relying solely on autogrowth.

Nutanix recommends starting with two TempDB data drives and one TempDB log drive. This arrangement should suffice for most SQL Server workloads.

Use trace flag 1118 to avoid mixed extent sizes. When using trace flag 1118, SQL Server uses only full extents. SQL Server 2016 automatically enables the behavior of trace flag 1118 for both TempDB and user databases. There is no downside involved in using this setting for SQL Server versions prior to 2016.

Windows Volume Configuration Best Practices

When formatting NTFS volumes for SQL Server data and log drives, set the allocation unit size to 64 KB.

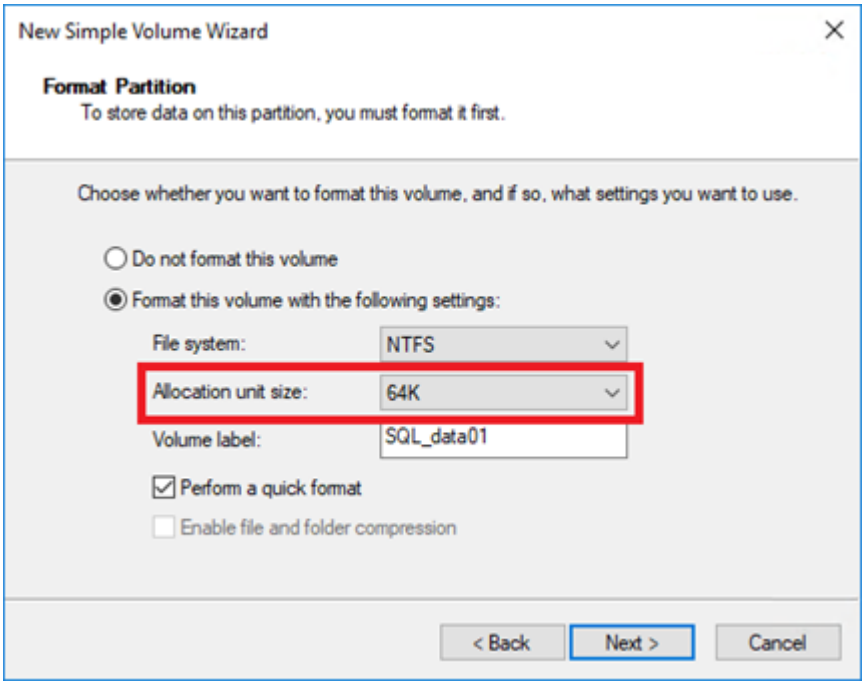

Figure 16: NTFS Allocation Unit Size

Microsoft SQL Server Filegroups

Applying Nutanix best practices to single-filegroup environments provides a good starting performance benefit. However, if a customer wants to isolate their metadata and user objects (non-system objects) or segregate different types of workloads in the same database into separate filegroups, there are also advantages to applying Nutanix storage recommendations at the filegroups level.

Through benchmarks and analysis of SQL Server filegroups, Nutanix has demonstrated that, as you increase the number of vDisks, you should also redesign your vDisk layout to accommodate the additional filegroups.

For more information, review the Microsoft SQL Server Filegroups tech note.

6. Microsoft SQL Server Design

Instant File Initialization (IFI)

When SQL Server data and log files are initialized, the process overwrites any existing data left on the disk from previously deleted files. Performing the following operations initializes data and log files:

- Create a database.
- Add data or log files to an existing database.
- Increase the size of an existing file (including autogrow operations).
- Restore a database or file group.

File initialization causes these operations to take longer.

In SQL Server, data files can be initialized instantaneously to avoid zeroing operations. Instant file initialization reclaims used disk space without filling that space with zeros, allowing for fast implementation of the file operations listed previously. Instead, disk content is overwritten only as new data writes to the files. Be aware that log files can't be initialized instantaneously, so correctly sizing the log files is a critical step.

Instant file initialization is enabled when the SQL Server service startup account has been granted the SE_MANAGE_VOLUME_NAME right. Add the SQL Server service account or the account used for backup operations to the Perform Volume Maintenance Tasks Windows security policy. Starting with SQL Server 2016, you can grant instant file initialization permission to the SQL Server service account during installation.

Nutanix recommends granting this right and enabling IFI. To validate that IFI is enabled, run the following T-SQL query:

```
$ SELECT servicename as Service_Display_Name,service_account as Service_Account,
 instant_file_initialization_enabled as Instant_File_Initialization_Enabled FROM
 sys.dm_server_services WHERE servicename LIKE 'SQL Server (%'
```


If IFI is enabled, the query returns the following result.

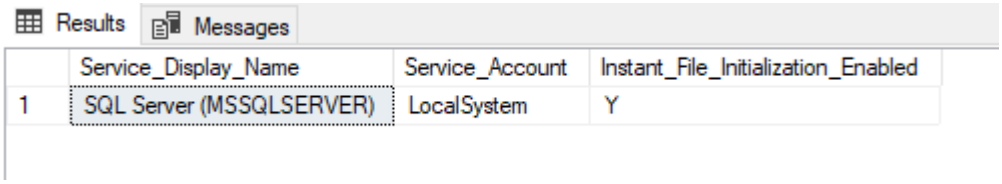

Figure 17: SQL Server IFI Query Result

SQL Server Memory Configuration

There are two important memory settings in SQL Server: maximum server memory and minimum server memory. Adjust these two settings in the Server Properties panel under Memory.

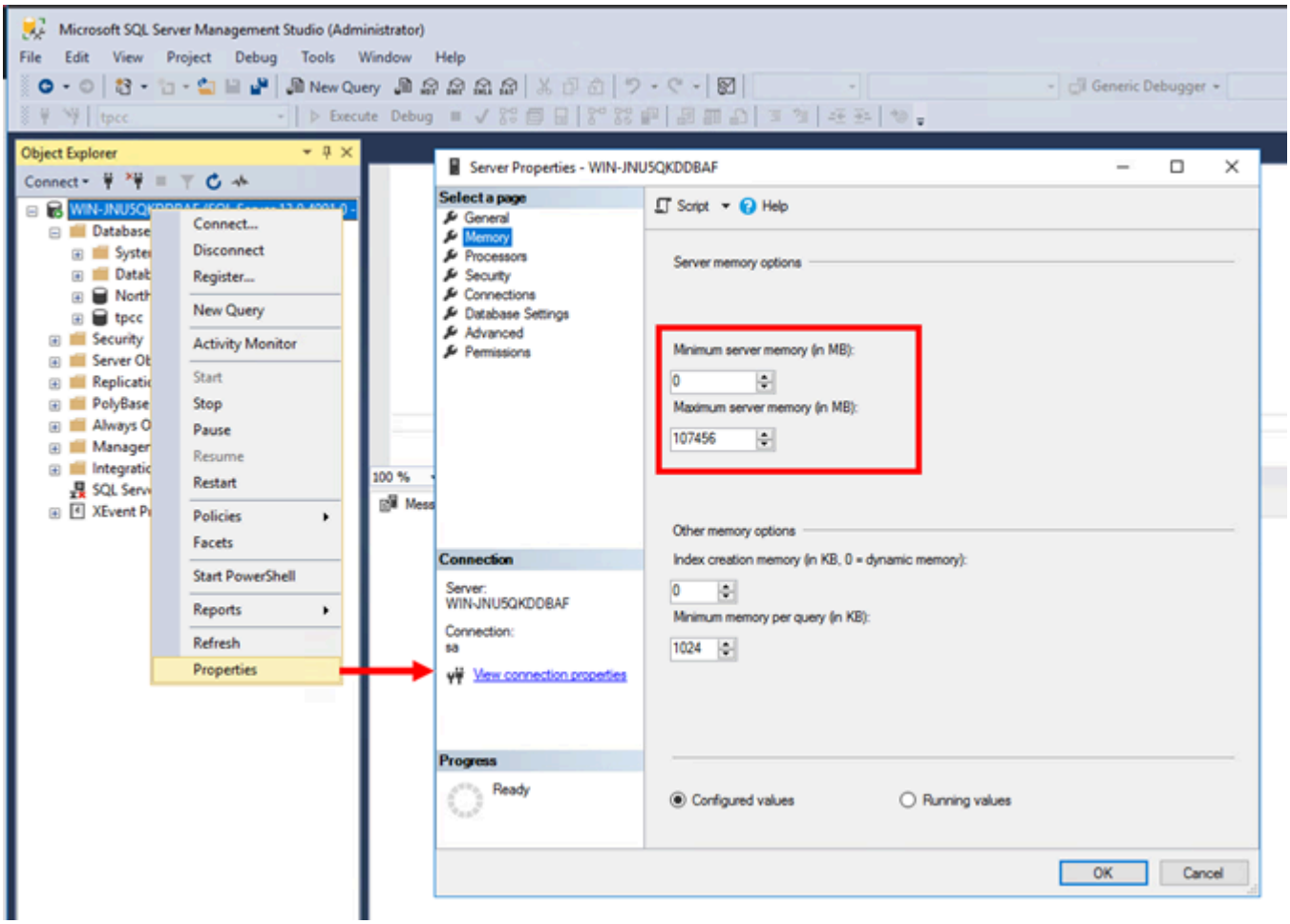

Figure 18: SQL Server Memory Settings

The default setting for maximum server memory is just over 2 PB (specifically, 2,147,483,647 MB). If you don't change this value, SQL Server tries to consume all the memory in the VM, which can have a negative impact on both SQL Server and the Windows OS.

As a general guideline, leave between 6 and 8 GB of memory for the OS and assign the remaining memory to SQL Server. If you need a more specific sizing guide, use the following formula to determine the optimal value for maximum server memory based on the VM configuration.

Maximum Memory = Total Physical Memory (MB)
minus OS Reserved (MB)
minus (250MB * Number of Cores)
minus (Number of SQL Threads * Thread Stack Size)

where: OS Reserved (MB):

5% of Total Physical Memory with a minimum of 1500MB and maximum of 4000MB.

Number of SQL Threads:

For 32-bit up to 4 processors, max worker threads equals 256.

For 32-bit with more than 4 processors, max worker threads equals 256 + ((Number of Cores – 4) * 8).

For 64-bit up to 4 processors, max worker threads equals 512.

For 64-bit with more than 4 processors max worker threads equals 512 + ((Number of Cores – 4) * 16).

Thread Stack Size:

2MB for 64-bit installations. 512KB for 32-bit.

Figure 19: Calculating SQL Server Maximum Memory

The following table is a quick reference for setting maximum server memory for a 64-bit OS.

Table: SQL Server Maximum Memory Settings

VM CPU	VM RAM	Max Memory
2	8 GB	4,976 MB
2	12 GB	8,976 MB
4	16 GB	12,476 MB
4	24 GB	20,476 MB
4	32 GB	28,376 MB
8	48 GB	42,448 MB
8	64 GB	57,648 MB

VM CPU	VM RAM	Max Memory
12	96 GB	87,720 MB
12	128 GB	119,720 MB

You must manually reconfigure SQL Server maximum memory any time the memory assigned to the VM changes. You must increase this parameter to allow SQL Server to take advantage of memory added to the VM, and you must reduce this parameter to leave enough memory for the Windows OS if memory is removed from the VM. When using large pages with the Lock pages in memory local security policy, restart the SQL Server service for the new value to take effect; this process incurs downtime.

SQL Server Performance Tuning

Several data points in SQL Server can indicate the performance efficiency of the database engine. One such data point is the CXPACKET wait type, which is a result of parallel query execution and indicates that a session is waiting for the synchronization of threads involved in the parallel process to complete.

If CXPACKET waits are more than 5 percent of the total waits for OLTP workloads, the environment could have a parallel processing problem. Keep these important considerations in mind when trying to resolve a high percentage of CXPACKET waits:

- If the query uses too many threads, the overhead of managing the threads can reduce the performance instead of improving it. To control the number of threads, use the maximum degree of parallelism or MAXDOP setting. You can define the MAXDOP setting on the instance, database, and query level. Setting the MAXDOP value to one disables parallel plans altogether. Setting the MAXDOP value to zero lets SQL Server use all available cores if a query runs in parallel.
- If small queries run in parallel, they consume more resources than if they ran serially. To control the greater expense of running parallel queries, you can tune the cost threshold for parallelism or CTFP setting. This setting adjusts the minimal cost for a query to be run in parallel. The cost is a custom SQL Server measure—it's not a measure of time.

When they see low performance on their database and the CXPACKET wait time is high, many DBAs decide to set the MAXDOP value to 1 to disable parallelism altogether. Because the database may have large jobs that could benefit from processing on multiple CPUs, Nutanix doesn't recommend this approach. Instead, we recommend increasing the CTFP value from 5 to approximately 50 to make sure that only large queries run in parallel. Set the MAXDOP according to Microsoft's recommendation for the number of cores in the VM's vNUMA node (no more than 8).

SQL Server Soft-NUMA

Microsoft introduced the soft-NUMA feature in response to the growing number of cores per CPU package. Soft-NUMA uses software to partition available CPU resources inside one physical NUMA node (hence the term "soft-NUMA").

With SQL Server 2016, whenever the SQL Server Database Engine detects more than eight physical cores per NUMA node or socket at startup, it automatically creates soft-NUMA nodes by default. Hyperthreaded processor cores aren't differentiated when counting physical cores in a node.

If using soft-NUMA doesn't provide a performance benefit for the workload, investigate whether you can change the vNUMA topology to expose smaller vNUMA to a VM hosting the instance of SQL Server.

SQL Server Compression Configuration

Data compression can save space and greatly improve performance. Microsoft introduced the ability to compress rows or pages in SQL Server 2008 and enhanced it in SQL Server 2012. Enabling compression can reduce the number of IOPS while allowing more data to fit in the Nutanix hot tier (SSDs), providing space savings as well as a performance boost. You can use compression with both OLTP and OLAP workloads, including data warehouse and batch processing.

Administrators can use compression technologies at either the SQL Server layer or the Nutanix storage layer. The results that compression can deliver vary based on the data in the database. It could be advantageous to enable

compression at both layers if the CPU cost-to-space savings are worth it. SQL Server and Nutanix use different algorithms to compress; you can enable Nutanix compression online without any downtime.

Lock Pages in Memory and Large Pages

The lock pages in memory (LPIM) Windows policy determines which accounts can use the LPIM process to keep data in physical memory, preventing the system from paging the data to virtual memory on disk. Locking pages in memory can keep the server responsive when paging memory to disk occurs. For 64-bit versions of SQL Server, Nutanix recommends testing the LPIM policy before pushing it to production.

For more information, review the Microsoft SQL Server and Windows LPIM Performance Analysis tech note.

Trace flag 834 causes SQL Server to use Microsoft Windows large-page allocations for the memory that's allocated for the buffer pool. With this flag, SQL Server allocates large pages at startup and keeps them throughout the lifetime of the process. Using large pages typically improves performance by increasing the efficiency of the translation look-aside buffer (TLB) in the CPU. You can use large pages if your environment meets the following conditions:

- The server is running SQL Server Enterprise Edition.
- The server has at least 8 GB of physical memory.
- The LPIM privilege is set for the service account.

Consider these factors if you plan to enable large pages:

- Large page allocation can increase SQL Server startup time. This approach changes the way SQL Server manages memory, allocating all buffer pool memory on startup instead of dynamically allocating it. Depending on the size of the instance, startup can take significantly more time.
- Nutanix doesn't recommend using large pages if you're using the COLUMNSTORE feature. See the Microsoft support article for more information on this issue.

Windows Guest Power Policy

In some cases, users may experience degraded overall performance on a Windows Server with SQL Server when running with the default (balanced) power plan. The issue may occur on any platform and in both native and virtual environments. The average response time for some tasks may increase, and CPU-intensive applications may experience performance issues.

Nutanix recommends using the high-performance power plan instead. This change allows the Windows scheduler to work most efficiently, as there is no dynamic scaling of CPU resources. Enabling this plan across all SQL Servers ensures a consistent level of performance.

High Availability Design

Microsoft provides a variety of high availability (HA) options to limit potential downtime. Nutanix recommends Always On availability groups (AGs) as the preferred HA solution. AGs are a shared-nothing model introduced with SQL Server 2012, where SQL Server can synchronously replicate transactions between nodes and provide very fast failover times. This approach doubles the amount of storage required when you use two replicas, triples the amount of storage when you use three replicas, and so on, as each node has a full copy of the database. AGs use a different framework than the traditional single-copy failover cluster, which requires shared SCSI disks. You can use hypervisor HA to help minimize server downtime as well.

If the application doesn't support the use of AGs, you can use log shipping as an alternative. Log shipping is an older technology, but it may have wider application compatibility. AGs are the modern and much preferred HA solution for SQL Server. If you can't use AGs or log shipping, Windows SQL Server FCIs are fully supported across all hypervisors if you use in-guest iSCSI to access a shared volume (as required for this type of solution).

Like other clustered applications, Windows failover clustering uses SCSI-3 persistent reservations to coordinate access to storage by enabling multiple instances of the application cluster to write to a shared drive. Prior to the AOS

5.17 release, administrators were forced to use iSCSI volumes to configure persistent reservations.

With AOS 5.17 and later versions, Nutanix makes it significantly easier to set up application-level clustering on AHV with volumes. The new functionality allows you to create a volume group and connect it to a VM using Prism, bypassing the need to configure iSCSI. The system can then access the volume group from within the guest as a SCSI device with access to persistent reservations.

AGs can also serve as a built-in disaster recovery solution, if you replicate an AG to a second or third datacenter. Ensure that you have sufficient bandwidth to keep up with the replication requirements and that latencies are acceptable.

When deploying FCIs or AGs, use hypervisor antiaffinity rules to enforce placement of the nodes on different physical hosts. This placement provides resilience in the case of an unplanned host failure. When designing your hypervisor clusters, allow for at least n + 1 availability. With this allowance, VMs can continue with adequate resources even if the system loses one host.

If using VMware HA, review the restart priority of VMs. You may want to increase the SQL Server priority to high for production databases. The best priority setting depends on what other services are running in your cluster.

Always On Failover Cluster Instance (FCI)

AGs are a great solution and fit nicely in the Nutanix scale-out architecture, but they may not be a good fit or even possible for certain environments. Some of the limiting factors for adopting AGs include:

- Space utilization: A secondary database copy consumes additional storage space. Some administrators may prefer a single database copy where server HA is the primary use case.
- Synchronous commit performance: Synchronously copying transactions (insert, update, delete) for AG replication (in the context of an HA solution) has a performance overhead. Administrators of latency-sensitive applications may prefer not to incur the additional response time of waiting for transactions to be committed to multiple SQL Server instances.
- Distributed transactions: Some applications perform distributed transactions across databases and SQL Server instances. Microsoft doesn't support

distributed transactions with AGs in SQL Server versions prior to SQL Server 2016 SP2, so application vendors can't support elements of those versions that use distributed transactions where AGs are present. Check Microsoft documentation for the most up-to-date information.

- Licensing: AGs require Enterprise Edition for SQL Server 2012 and 2014. Some cost-conscious organizations may prefer to use Standard Edition for many of their SQL Server instances but still desire high availability at the SQL Server level. SQL Server 2016 and newer provide support for basic AGs with Standard Edition with several limitations.

In these cases, SQL Server Failover Cluster Instances (FCIs) can offer a solution. FCIs have long been used as a means for HA with SQL Server. FCIs work with all current versions of SQL Server and rely on shared storage to support the SQL Server instances.

The shared storage used can be block (LUN) based or, starting with SQL Server 2012, SMB (file) based. In the case of LUN-based shared storage, SCSI-3 persistent reservations arbitrate ownership of the shared disk resources between nodes. Refer to Microsoft's documentation for additional details on Always On FCIs.

Nutanix supports the use of FCIs across ESXi, Hyper-V, and AHV if you use in-guest iSCSI. With AOS 5.17 and later versions, customers using AHV as the hypervisor can simply use direct-attached volume groups, bypassing the need to use in-guest iSCSI LUNs for FCIs and other clustering applications. Hyper-V environments can also take advantage of the shared virtual hard disk functionality introduced with Windows Server 2012 R2.

Backup and Disaster Recovery Design

It's extremely important to meet recovery point objectives (RPOs) and recovery time objectives (RTOs) in a SQL Server environment. Depending on the business requirements, these targets may be very stringent. Sizing SQL Server VMs must take RTOs into account. For example, a massive SQL Server VM takes much longer to restore than a smaller VM. If your backup subsystem can restore at the rate of 1 TB per hour, and your RTO for a tier-1 SQL Server VM is one hour, the VM should be less than 1 TB.

The additional time necessary to restore massive VMs is one reason that Nutanix recommends scaling out SQL Server VMs rather than scaling up. Work with your backup team to establish backup policies that meet RTOs and RPOs for each database.

As previously discussed, SQL Server AGs are an excellent built-in disaster recovery solution for your databases. Ensure that there is adequate bandwidth between the sites to sustain the replication traffic.

SQL Server Maintenance Scripts

Maintaining your SQL Server is very important. For a comprehensive set of backup, integrity check, index, and statistics maintenance scripts check out Ola Hallengren's scripts.

Another handy script for checking the health of SQL Server is sp_Blitz by Brent Ozar.

Both solutions are free; carefully evaluate them for use in your environment. They can automate much of the drudgery of maintaining SQL Server databases.

In a scale-out infrastructure, to make use of the architecture, use multiple I/O data streams and multiple backup streams. Here's an example script that configures a backup of a SQL Server DB to write to multiple files in a single drive:

```
$ BACKUP DATABASE [ntnxdb] TO
$ DISK = N'R:\SQLBACK\ntnxdb-tpcc-SQL2014-1000wh-01.bak',
$ DISK = N'R:\SQLBACK\ntnxdb-tpcc-SQL2014-1000wh-02.bak',
$ DISK = N'R:\SQLBACK\ntnxdb-tpcc-SQL2014-1000wh-03.bak',
$ DISK = N'R:\SQLBACK\ntnxdb-tpcc-SQL2014-1000wh-04.bak'

$ WITH  DESCRIPTION = N'ntnxdb',
 NOFORMAT, NOINIT,  NAME = N'ntnxdb,
 SKIP, NOREWIND, NOUNLOAD, COMPRESSION,
 STATS = 5
$ GO
```

For larger databases, using multiple drives to host those backup files can greatly improve performance. An effective (and simpler) way to manage multiple drives is to use mount points and assign each drive to a folder rather than to a letter.

In the following example script, the SQLBACK1, SQLBACK2, SQLBACK3, and SQLBACK4 folders serve as mount points for four independent disks. Nutanix recommends using multiple backup files to benefit from the Nutanix scale-out architecture.

```
$ BACKUP DATABASE [ntnxdb] TO
$ DISK = N'R:\SQLBACK1\ntnxdb-tpcc-SQL2014-1000wh-01.bak',
$ DISK = N'R:\SQLBACK2\ntnxdb-tpcc-SQL2014-1000wh-02.bak',
$ DISK = N'R:\SQLBACK3\ntnxdb-tpcc-SQL2014-1000wh-03.bak',
$ DISK = N'R:\SQLBACK4\ntnxdb-tpcc-SQL2014-1000wh-04.bak'

$ WITH  DESCRIPTION = N'ntnxdb',
 NOFORMAT, NOINIT,  NAME = N'ntnxdb,
 SKIP, NOREWIND, NOUNLOAD, COMPRESSION,
 STATS = 5
$ GO
```


7. Best Practice Checklist

The following is a consolidated list of all the best practices covered in this document.

General

- Perform a current state analysis to identify workloads and sizing.
- Spend time up front to design and build a solution that meets both current and future needs.
- Design to deliver consistent performance, reliability, and scale.
- Don't undersize, don't oversize—right size.
- Start with a proof of concept, then test, optimize, iterate, and scale.

Drive Layout and Configuration

- Use separate drives for the OS and SQL Server binaries.
- Ensure that user databases and system databases are on separate drives.
- Use one drive for database log files and another drive for the TempDB log file.
- Distribute databases and log files across multiple vDisks.
- Distribute vDisks across multiple SCSI controllers.
- Use the VMware PVSCSI controller for database and log disks or for all drives if possible.
- Use 64 KB NTFS allocation for database and log drives.
- Size for at least 20 percent free disk space on all drives.
- Don't use Windows dynamic disks or other in-guest volume managers.

SQL Server Data Files

- Enable instant file initialization (IFI).
- For each database, use multiple data files: one file per vCPU.
- For very write-intensive databases, distribute database files over four or more vDisks.
- Size all data files in a file group equally.
- Enable autogrow in 256 MB or 512 MB increments to start.
- Use trace flag 1117 for equal file autogrowth. With SQL Server 2016 and newer, use the ALTER DATABASE AUTOGROW_ALL_FILES command instead.
- Don't shrink databases.

SQL Server Log Files

- Base total size on the high-water mark before the next log truncation period.
- Optimal VLF size is between 256 MB and 512 MB.
- No log file should exceed 10,000 VLFs.
- Configure initial log file size to 4 GB or 8 GB and iterate by the initial amount to reach the desired size.
- DBCC LOGINFO ("dbName"): The number of rows returned is the number of VLFs.
- One log per database (including TempDB) is sufficient, because SQL Server can only use a single log file at any given time.

TempDB

- Use multiple TempDB data files, all the same size.
- Use autogrow on TempDB files with caution to avoid situations where files use 100 percent of the disk that hosts the log files.

- If there are eight or fewer cores, the number of TempDB data files is equal to the number of cores.
- If there are more than eight cores, start with eight TempDB data files and monitor for performance.
- Initially size TempDB to be 1 to 10 percent of database size.
- Use trace flag 1118 to avoid mixed extent sizes (full extents only). SQL Server 2016 automatically enables the behavior of trace flag 1118.
- One TempDB log drive should be sufficient for most environments.

VMware

- Use the VMXNET3 NIC.
- Use the latest VMware VM hardware version.
- Use the PVSCSI controller when possible.
- Remove unneeded hardware (floppy drive, serial port, and so on).
- Don't enable CPU hot-add, as this disables vNUMA.

RAM

- More RAM can increase SQL Server database read performance.
- Enable large page allocations when using 8 GB of memory or more.
- Don't overcommit RAM at the hypervisor host level.
- For tier-1 databases, reserve 100 percent of RAM.
- Configure SQL Server maximum memory.
- For tier-1 workloads, lock pages in memory.
- Size each VM to fit within a NUMA node's memory footprint.

vCPUs

- Don't overallocate vCPUs to VMs.
- For tier-1 databases, minimize or eliminate CPU oversubscription.
- Account for Nutanix CVM core usage.
- If possible, size VMs to fit within one NUMA node.

Networking

- Use hypervisor network control mechanisms (for example, VMware NIOC).
- Use VMware load-based teaming with the vDS.
- Connect Nutanix nodes with redundant 10 Gbps connections.
- Use multi-NIC vMotion or Live Migration.

Power Policy

- If using Hyper-V hosts, use the high-performance power option.
- If using VMware hosts, use balanced power mode unless the database is highly latency sensitive.
- Always set the Power policy to High Performance in the VM guest OS.

SQL Server Instance Scaling

- To make the most of a scale-out platform, configure the SQL DB to use multiple files across multiple vDisks.
- We don't recommend scaling up one VM with multiple SQL Server instances.

High Availability

- Use SQL Server Always On availability groups for SQL Server HA.

- Leave VMware DRS automation at the default level (3).
- Let the hypervisor and DRS manage noisy neighbors and workload spikes.
- Use at least n + 1 for HA configuration.
- Use hypervisor antiaffinity rules to separate FCIs and AG cluster nodes.
- Use a percentage-based admission control policy for VMware environments.
- Review SQL Server VM restart priorities.

Monitoring

- Choose an enterprise monitoring solution for all SQL Servers.
- Closely monitor drive space.

Manageability

- Standardize on SQL Server build and cumulative updates.
- Use standard drive letters or mount points.
- Use VM templates.
- Join the SQL Server to the domain and use Windows authentication.
- Use Windows cluster-aware updating for AG instances.
- Test patches and roll them out in a staggered manner during maintenance windows.

SQL Server Compression

- Carefully evaluate using SQL Server compression for all databases.
- Look into using SQL Server compression either at the row or page level, depending on which one provides the best savings based on the workload. Be aware of licensing.

Nutanix Recommendations

- Enable inline compression (delay set as zero) at the container level, as this setting benefits sequential I/O performance. If your environment has high CPU usage (consistently over 80 percent on the CVM) or if the environment is highly sensitive to latency, we recommend setting compression with a 60-minute delay to avoid using too many CPU cycles.
- Don't use container-level deduplication.
- Only use Nutanix erasure coding (EC-X) after carefully examining the workload pattern, because EC-X is best suited to datasets that aren't written or rewritten often. Reach out to the Nutanix account team for further guidance.
- Use the fewest possible containers (one is fine).
- Take database growth into account when calculating sizing estimates.

Backup and Disaster Recovery

- Establish RTOs and RPOs for each database.
- Validate that the backup system can meet RTOs and RPOs.
- Size SQL Server VMs so they can be restored within RTOs.
- Use AGs as a built-in disaster recovery solution.

SQL Server Maintenance

- Use Ola Hallengren's SQL Server maintenance scripts.
- Use Brent Ozar's SQL Server sp_Blitz health check script.

8. Conclusion

Microsoft SQL Server deployments are crucial to organizations, as they're used in everything from departmental databases to business-critical workloads, including ERP, CRM, and BI. At the same time, enterprises are virtualizing SQL Server to shrink their datacenter footprint, control costs, and accelerate provisioning. The Nutanix platform provides the ability to:

- Consolidate all types of SQL Server databases and VMs onto a single converged platform with excellent performance.
- Start small and scale databases as your needs grow.
- Eliminate planned downtime and protect against unplanned issues to deliver continuous availability of critical databases.
- Reduce operational complexity by leveraging simple, consumer-grade management with complete insight into application and infrastructure components and performance.
- Keep pace with rapidly growing business needs, without large up-front investments or disruptive forklift upgrades.

9. Appendix

Resources

SQL Server Licensing

1. Microsoft product licensing for SQL Server

Supporting SQL Server on Nutanix

1. Nutanix third-party hardware compatibility lists
2. Nutanix Compatibility and Interoperability Matrix (including guest OS options supported on Nutanix AHV)
3. VMware Compatibility Guide (including guest OS options supported on VMware ESXi)
4. Supported Windows guest operating systems for Hyper-V on Windows Server
5. Nutanix Software Documents (including release notes)
6. Nutanix End of Life Information
7. Support policy for Microsoft SQL Server products that are running in a hardware virtualization environment
8. Microsoft Lifecycle Policy (to validate that your versions of SQL Server and Windows Server are in support)
9. Server memory configuration options

Nutanix Networking

1. Nutanix AHV Networking best practice guide
2. VMware vSphere Networking best practice guide
3. Hyper-V Windows Server 2016 Networking best practice guide

About the Authors

Bruno Sousa is a Technical Director for Database Solutions at Nutanix and a Nutanix Platform Expert (NPX-015). Follow Bruno on Twitter at @bsousapt.

Shankar Govindan is a Solutions Architect in the Business-Critical Applications Engineering team at Nutanix.

Chris Jones is an Advisory System Engineer in the Enterprise team at Nutanix. Follow Chris on Twitter at @cpjones44.

Michael Webster is a Principal Solutions Architect in Nutanix Engineering and a Nutanix Platform Expert (NPX-007). Follow Michael on Twitter at @vcdxnz001.

About Nutanix

Nutanix offers a single platform to run all your apps and data across multiple clouds while simplifying operations and reducing complexity. Trusted by companies worldwide, Nutanix powers hybrid multicloud environments efficiently and cost effectively. This enables companies to focus on successful business outcomes and new innovations. Learn more at Nutanix.com.

List of Figures