SAS1931-2018

Deploying SAS® Grid Systems on VMware ESXi Virtually Provisioned Storage

Tony Brown, SAS Institute Inc.

ABSTRACT

Creating performant SAS® file systems that are compliant with VMware ESXi can be a challenge. There are several ways to implement virtual and software-defined storage resources within a VMware ESXi hypervisor controlled system. This paper covers several considerations for file system and logical unit number (LUN) construction from several perspectives: host logical volume, virtual machine file system (VMFS) and virtual machine disk (VMDK), and VMFS and raw device mapping (RDM) implementations. Also discussed are non-LUN mounts to hyper-converged storage systems. The focus is on tuning and performance of the method used, as well as pros and cons between hypervisor-level, and back-end storage-level utility operations.

INTRODUCTION

Many SAS® Grid Computing implementations running on VMware ESXi virtual systems have encountered performance issues and ended up requiring technical and performance support. The underlying causes typically involve insufficient physical provisioning beneath the virtual environment or improper virtual-to-physical implementation and tuning settings. This can occur as a result of sub-optimal virtual host and storage definitions and tuning, underlying physical host provisioning, and physical network (network interface cards [NIC], network cabling, and switches) bandwidth between the host nodes and bandwidth in connections to back-end storage. Tuning and provisioning along each step of the infrastructure at both the physical and virtual layers is crucial to providing adequate performance for the heavy resource demands of SAS®.

One of the pre-eminent goals of virtualization is to easily create, size, and move resources within commodity pools of hardware while driving utilization to 100%. The software-driven approach to accomplish this relies on creating virtual levels of abstraction from large, thin, physical pools of host and storage resources. This often leads to provisioning decisions of virtual-on-physical that countermand best practices for SAS. The CPU, memory, and IO bandwidths demanded by SAS usually require available, often non-shared, thick physical resources. That is a substantially different requirement than can always be met by commoditizing virtual spaces from large, shared pools.

This paper addresses best practices for implementing virtual machine (VM) systems and virtualized host and storage resources, while still enabling sufficient physical bandwidth for adequate performance. There are some tradeoffs between the ideal “thin” definitions of the abstracted virtual resources and the less ideal “thicker” assignment of underlying physical resources. This paper discusses those tradeoffs. The general storage concepts given can be applied to your storage paradigm with help from your vendor engineer.

SAS® Metadata Server and SAS middle-tier servers typically function and perform well in virtualized environments. The performance challenges of virtual provisioning show up in the SAS Grid compute nodes. A previous paper details some general best practices for moving SAS Grid systems to VMware ESXi hosts. While the advice in that paper is still largely applicable, some of the advice is updated in this paper, along with newer and more detailed tuning and provisioning recommendations. The previous paper can be found here:

http://support.sas.com/resources/papers/MovingVirtuaVMware.pdf

This paper starts at the physical host layer with recommended BIOS settings then explores physical host considerations, virtual machine configurations, general storage guidelines, and finally, logical unit number (LUN) provisioning methods for performance.

Highly generalized guidelines have been published for VM hosts and are updated with additional material.

BIOS SETTINGS

There are two important BIOS settings that require attention for nodes that host VMs:

1) Enable hardware virtualization.
2) Disable node interleaving. While enabling node interleaving is favored by some installation precepts, it can result in long memory accesses, sometimes with a poor locality of reference.

See the following link for help with changing BIOS settings: https://docs-old.fedoraproject.org/en-US/Fedora/13/html/Virtualization_Guide/sect-Virtualization-Troubleshooting-Enabling_Intel_VT_and_AMD_V_virtualization_hardware_extensions_in_BIOS.html

PHYSICAL HOST CONSIDERATIONS

The foundation of any system ultimately lies in the architecture and tuning of its physical hardware. There are some specific considerations to make to ensure that the hardware is crafted to perform well. This section addresses those items. SAS provides tuning guides for its compatible host operating systems. These guides are written for specific operating systems and versions. They provide memory management, cache, IO, file system, logical volume tuning, and IO multipath configuration guidance. These precepts should be followed at the Guest operating system (OS) level even on virtual machines. Consult with your VMware architect to ensure any hypervisor-level tuning will not supersede or countermand the tuning recommended in these host guides. These tuning guides can be found here: http://support.sas.com/kb/53/873.html

The following link contains many papers that are associated with the above, offering guidance on general performance precepts, storage, shared file systems, cloud and converged infrastructures, and so on. These papers can be found here: http://support.sas.com/kb/42/197.html

In addition to the Guest OS tuning from the papers above, there are CPU, memory, and IO-specific considerations to be made on the physical servers that underlie VM installations. Those items are addressed below.

Memory – When installing or adding to memory in a physical host, it is preferable to install DIMMs in groups of 6 across current dual socket server types, and balance the DIMMs in number across the sockets. This provides uniform capacity per socket and helps with memory reach. Shorter memory reaches mean faster service. The fastest memory affordable should be used (for example, fastest GHz rating). Heavy SAS analytic workloads can be very memory-intensive and faster memory significantly helps.

CPU – It is important that processor power management and sleep states are set appropriately. A P-State is a power management setting (Advanced Configuration and Power Interface [ACPI]) that determines the power level (P-State) the CPU is running at. Most CPU vendors define between 10 and 12 P-States for their CPUs, and these P-State levels are processor-specific. These values represent the amount of power the chip can consume (from the maximum to the least) and the ensuing GHz speed required by the power level (maximum to least). Table 1 below shows the Intel E5 2930 V4 Chip has 12 P-States, ranging from the fully powered up P0 State, consuming the most electricity (running at 85 watts and 2201 MHz clock speed) to the lowest power state of P11 (running at 20 watts and 1240 MHz clock speed). While turning P-States down will save power, it also degrades SAS performance considerably. We highly recommend a P-State setting of P0 for SAS systems.

P-State	Frequency	Thermal Dynamic Power Rating
P0	2201 MHz	85 W

P1	2200 MHz	85 W
P2	2100 MHz	79 W
P3	2000 MHz	74 W
P4	1900 MHz	70 W
P5	1800 MHz	65 W
P6	1700 MHz	61 W
P7	1600 MHz	57 W
P8	1500 MHz	53 W
P9	1400 MHz	48 W
P10	1300 MHz	45 W
P11	1200 MHz	40 W

Table 1. P-State Settings for the E5 2930 V4 Processor (P-States Vary by CPU Model)

While P-States govern power-level consumption and the resulting clock speed, another setting, the C-State, governs the "sleep" or "idle" state the processor drops into when it is not fully active. Intel provides four C-States:

- C0 – CPU is always fully powered up and running. No sleep. High power consumption.
- C1 – CPU enters a halt state but remains awake and servicing cache coherence events, for example. Light sleep, aware of what is going on though not actively working. 2 µs minimum time to wake up.
- C1E – CPU is voltage ready, but is at the C1 state and the lowest P-State for power consumption. Same sleep as C1, but power has been turned to lowest setting (P-State 11 on an E5 2930 V4). 10 µs minimum time to wake up.
- C3 – CPU has internal clocks stopped and does not service cache coherence events. Active L1 and L2 cache flushed into last-level cache. Deep sleep, takes longer to wake up and resume production. 40 µs minimum time to wake up and re-establish L1 and L2 cache.
- C6 – CPU is in deepest sleep, consuming 0 volts. All cache is flushed to static RAM and the CPU must completely power up and re-establish all cache to resume work. Longest time to wake up and resume production. 133 µs minimum time to wake up and re-establish all cache.

Most modern CPU models support Intel Turbo Boost (a quick increase in speed) when needed. Turbo Boost works by borrowing power, thermal capacity, and so on from less used CPUs. Turbo Boost typically requires a C-State of 3 or less and can increase the speed, on average, from 0.7 to 1.3 GHz on an E5-2650 V4, for example. This is very significant. Lab testing shows Turbo Boost can make a significant short-term difference on individual floating point and integer processing intensive jobs.

Traditional OS tuning guides on bare-metal servers (not virtualized) recommend configuring BIOS settings by using OS tuning profiles that specify maximum power (highest P-State) and a C-State that does not include deep sleep (for example, a C-State no deeper than C1 or C1E, if supported). Heavily used, round-the-clock production machines tend to suffer high start-up latency from lower power levels and deeper sleep states.

VMware highly recommends performing all power management for P-State and C-State settings through the VMware vSphere power management system. They do not want vendor-specific BIOS profiles set on the physical servers. The reasons for this are widespread maintenance management of hosts, the ability to make power management changes without requiring reboots (BIOS settings), and the ability to adjust power settings on the fly as needed by the environment.

Here are the VMware recommendations:

1) Set the High Performance policy in vSphere.
2) Disable Turbo Boost via vSphere only if needed (enabled in the BIOS).
3) Set ***PowerMaxFreqPCT*** to .99 to keep the CPUs at high power.

Whether you choose to make BIOS setting changes on your ESXi hosts via vSphere or via the Guest OS, setting maximal power and availability are crucial.

Broad field performance experience on both virtualized and non-virtualized systems shows that SAS workloads have the best overall performance experience with a combination of maximum power and high C-State availability, combined with Turbo Boost. The benefit from Turbo Boost in most workloads is significant, as is the penalty from waking processors up from a deeper C-State sleep to enable it. SAS workloads are highly resource impactful, with aggressive SLAs, and unpredictable ramp-up in host CPU resource demand. We do not recommend adjusting these settings over time for our workload, nor changing them on the fly for environmental reasons. (Everything we do requires very thick provisioning and immediate availability of full CPU power and resources.)

It is our recommendation to use the highest P-State and C-State settings, while enabling Turbo Boost. You must experiment with your own system to determine the best P-State and C-State balances for your environment, workload, and customer performance SLAs.

IO – Communication card ports should use dedicated application-specific integrated circuit (ASIC) chips (not shared). This helps ensure a fuller physical bandwidth. In addition, the Fibre Channel cards (host bus adapter, for direct attachment) or network interface cards (for LAN attachment) should be chosen to match the wire speed of the underlying network fabric and switches. Otherwise, the cards can become a bottleneck. Bonding NIC cards is a common practice to aggregate card bandwidth.

It is important to enable offloading in NIC cards to ensure maximum performance. NIC offloading allows the physical NIC card to handle network- or protocol-specific processing of shipped segments instead of the host CPU. This places less strain on the host system. NIC offloading works fairly well if your NIC cards support large segment offload (LSO) and large receive offload (LRO) for outbound and inbound packet segments, respectively. If supported, LSO/LRO is enabled by default in your physical NICs, in the VMkernel layer and in the VMXNET3 VM adapter. However, this setting might not be enabled for the Guest OS and must be checked!

Note: To determine if your physical NIC supports TCP segmentation offload (TSO) and if it is enabled by your ESXi host, run the ***excli network nick tso get*** command.

The Guest OS guides recommend the use of round robin multi-pathing for IO placement across storage connections and volumes. We have tested various native multi-pathing and vendor-specific multi-path software in controlled workload and hardware environments. We have found no considerable difference between their overall IO performances. Use the multi-path software you are most familiar and comfortable with.

As an engineering exercise, it is crucial to assess the IO path from the OS stack through the adapter cards and the fabric and switches of the network to the device layer of the back-end storage. This path includes the Guest OS and VMware vSphere Hypervisor provisioning, configuration, and settings. This ensures that all aspects of a full bandwidth pipe are identified and correctly set in order to achieve the required bandwidth of a sustained 125 megabytes per second (MB/s) per SAS CPU core used in each VM.

When configuring host bus adapters, it is sometimes helpful to modify the adapter policy in both the Guest OS and VMware to set the ***IO Throttle*** to 1024. This can vary by adapter, so please refer to the documentation for the specific model you are using.

Finally, it is important not to establish host IO limits to SAS storage groups. SAS is often used in an ad hoc manner and workload bandwidth can spike precipitously at times. Enforcing IO limits can put a ceiling on performance.

VIRTUAL MACHINE GENERAL CONFIGURATION GUIDANCE

When configuring a VM host there are several provisioning details that are important to maximize CPU, memory, and IO performance.

CPU AND MEMORY

CPU core count. When planning core capacity for VMs, be aware that the cores are not exactly equivalent from a capacity perspective. This is due to VM hypervisor overhead usage of system cores. An 8-core physical SAS Grid node is equivalent to approximately a 7½-core virtual host. This jumps to one full core difference on a 16-core node: 16 physical cores ~= 15 VM cores.

CPU and memory overhead. It is crucial to build in overhead when planning CPU and memory capacity for a VM. In VM hosts, it is highly advised to build in an additional 15% overhead for both CPU and memory allocation to allow for hypervisor-related services. When overall memory utilization violates this 15% threshold boundary, it is time to add memory. The shared file system you use can bump this overhead margin to 20%.

CPU and memory placement. It is highly advised to restrict CPU cores and memory for a single VM to within a single physical host socket. It is also advised to allocate one VM core directly to 1 physical core. (For example, do not split VM cores across physical cores.) This helps avoid non-uniform memory access (NUMA) issues. It is helpful to use the ***esxtop*** memory monitor, looking for the *%local* variable to determine memory locality of reference. Poor locality of reference adds unwanted latency to memory operations.

It is also necessary to use a persistent allocation of CPU and RAM for VM provisioning that is fully reserved. The reason for this thick provisioning is the latency cost of re-obtaining over-subscribed resources.

IO

For network attached storage, **VMXNET3** VM network adapters should be used if your NIC cards support LSO/LRO. (See the IO section under “Physical Host Considerations,” above.) In addition to VMXNET3 adapters, the following will allow NIC sharing, even over InfiniBand fabric:

- **VMDIRECTPATHIO.** It is helpful to set up NIC pass-through on fully allocated NICs using VMDirectPath IO. This will disable vMotion capabilities, however, so beware of that drawback!
- **SRIOV**. Single Root IO Virtualization (SRIOV) is a method to allocate a single channel of a NIC to a VM host while still sharing the other channels. Use SRIOV on approved NICs Models only. See the guidelines and advice in the vSphere Compatibility Guide for Virtual SAN. This online tool is available at: https://www.VMware.com/resources/compatibility/search.php?deviceCategory=vsan.

For direct-attached storage:

- **PVSCSI**. It is recommended to use the ParaVirtual Small Computer Systems Interface (PVSCSI) controller option for adapters. PVSCSI controllers have tunable queue depth settings, which typically default to 64. This setting must be raised substantially (the maximum is 1024) to at least match queue depths of your direct-attached controllers. For example, if you use internal host drives to stripe SASWORK/UTILLOC, you want to match the PVSCSI queue depth setting with the hardware RAID controller, and perhaps even to the 1024 maximum. The PVSCSI controller might require additional Guest OS drivers to be installed. A VM can support a maximum of 4 PVSCSI adapters, with 2 volumes per adapter. With solid state disk (SSD) back-end storage on an 8-core SAS Grid node, one adapter is typically sufficient, but your experience can vary based on the workload and file systems over different LUN architectures.
- **NVMe.** When using NVMe-based storage, the Virtual NVMe (**vNVMe**) storage controller allows processing on multiple storage queues in parallel (up to 64 queues with a queue depth of 64 on

each). It is available starting in VM hardware version 13. With up to 4 vNVMe controllers in each VM host, this can help provide much higher channel parallelization, combined with lowered latency.

- **VDT**. The recommended Virtual Disk Type (VDT) is **Thick Eager Zeroed Disk Provisioning**. This virtual disk provisioning method pre-allocates the disk space to the VMDK and zeros it out upon creation. Thickly provisioned assets can provide marginally better performance, even with the much improved automated reclamation in ESXi 6.5 and later releases. Some storage arrays (for example, EMC) don't write zeros to physical storage rendering that benefit moot. Work with your storage vendor to understand their particular array behavior with zero writes and determine whether the balance of benefits of thick versus thin provisioning isn't ameliorated by the absence of zero write behavior.
- **VAAI.** When using enterprise level SSDs, it is important to enable vSphere Storage APIs for Array Integration (VAAI). This enables the Linux UNMAP commands, which support block un-mapping on flash storage (without VAAI, these commands do not work on virtual disks in ESXi6). The VMkernel will convert its TRIM function to the UNMAP commands supported by Linux for enterprise SSD operations. This allows more efficient block erasure functions, and mitigates some garbage collection impact on the flash devices. It is important to note for VAAI that UNMAP is not supported for vSAN versions 6.6 and prior.
- **vSAN.** The subject of vSAN is lengthy and complex, and beyond the scope of this paper. Some basic recommendations are to carefully provision the vSAN caching disk and to pay close attention to replication policies and locality of reference when establishing disk groups across the node cluster.
    - The best underlying hardware is high-speed SSDs.
    - When using vSAN, it is important to turn off the deduplication and compression feature. The latency added to SAS workloads from this feature is considerable.
    - A high-speed fabric interconnect is crucial. Replication Writes in the erasure coding schemes cannot be disabled and they pose a serious added burden to the fabric for the heavily used and data-amplified scratch space in SASWORK and UTILLOC.

GENERAL SAN STORAGE GUIDANCE

ARRAY STORAGE – DIRECT ATTACHED

Listed below are an array of items to pay attention to. They are not listed in an order of precedence, but occur from the physical and virtual host storage access definitions through the back-end storage.

Physical

When using direct-attached storage with host bus adapters (HBAs), it is important to use enough cards and bandwidth to supply each host with a minimum of 125 MB/s per SAS CPU core throughput. For example, a typical 8-core SAS Grid node would minimally require 8 cores x 125 MB/s per SAS CPU core = 1 GB/sec throughput to storage. This would be right at the ceiling of a single 8 Gbit host bus adapter. To allow for additional overhead, it would be wise to use a dual port adapter or more cards.

When using Fibre Channel ports on storage, it is important to not use a shared ASIC for a port unless that port is configured for full throughput delivery. Doing so can introduce a bottleneck. In addition, queue depths should be as synonymous as possible across the entire Fibre Channel link, from the host to the array and to resulting LUNs. Typically, minimal HBA and storage array LUN queue depth settings should be at least 256. This figure can be array card type and model-specific, so check with your storage vendor engineer.

Whether you are deploying internal node drives or external-to-the-node, direct-attached SAN for SASWORK, you must maximize bandwidth to the devices. This means doing the math on what you are provisioning in order to attain sufficient MB/s per SAS CPU core for the host nodes. This starts with the host initiators and hardware RAID controllers and continues to the connectivity and the storage devices.

For SAN, the Fibre Channel connections, storage front and back-end adapter ports, controller bandwidth, device pool, and so on must all be replete enough to carry the total MB/s bandwidth.

It is important to understand array services and the resources they require. They can interfere enough with array operations and bandwidth to impact running workloads. These include compression, de-duplication, backups, replication services, and so on. See the following link for more information: http://support.sas.com/resources/papers/proceedings15/SAS1500-2015.pdf

Additional VM Host Settings That Impact Direct-Attached Storage

From the virtual host perspective, it is advisable to increase the number of consecutive requests from a single VM from 8 to 64 in vSphere, especially with large-sized LUNs (for example, >1TB).

When setting total storage queue depth targets from a VM host, the actual queue depth can vary by storage offering. For example, in the EMC family, the XtremIO and Unity queue depth target is 1500, VMAX 3 is 4,000, and DSSD is 64,000. Work with your storage vendor to plan queue depth capacity and targeting from the virtual host through to the storage array and the underlying LUNs in the device pool. The targets can be substantially different for dual-controller storage architectures versus virtual machine interconnect architectures.

The precepts in the IO section under “Virtual Machine General Configuration Guidance” above must be followed.

ARRAY STORAGE – NETWORK ATTACHED

In addition to the precepts in the IO section under “Virtual Machine General Configuration Guidance” above, physical bandwidth must be provided for network attached storage. This includes sufficient bandwidth from the NIC cards, network fabric, and switches. A minimum of 125 MB/s per SAS CPU core must be garnished through the entire chain.

A very important note for SAS Grid is to not provision the network fabric bandwidth too tightly, especially when sharing the same NIC arrangement as the local node interconnect. The local node interconnect is used by both the SAS® Grid Manager, metadata, and scheduling services, as well as the shared file system (GPFS, GFS2, and Veritas, for example) to maintain communication for node coherency, lock management, and shared file system services. If the fabric on which this takes place is bandwidth constrained and causing communication delays, the SAS Grid can lose quorum with its servers and the shared file system can fence unresponsive nodes. This is not just a performance issue, but one of stability of the running application.

SCALE OUT (DEVICES IN-NODE) STORAGE

The promises of scale-out, software-defined storage (SDS) in hyper-converged infrastructures (HCI) abound. They espouse the benefits of elastic storage—rapid spin-up and spin-down and the ability to, create, modify, manage, and destroy on the fly—all from a software control panel governing commodity pools of physical storage. This thin, shared, over-subscribed storage pool is the way of the future. The VM hypervisor wants full control of not just the VM hosts and infrastructure, but also the storage. With this full control, elastic growth, policy provisioning, and rapid management, it seems possible to eliminate the extensive maintenance costs, bottlenecks, and inefficiency of the old monolithic SAN world.

However, I have three personal axioms:

- There is no such thing as magic.
- There is no free lunch.
- You cannot change the laws of physics with the amount of money you are spending.

I do fully support the concepts, ideas, and goals of HCI and scale-out software-defined storage (SDS). There are always smarter ways to approach problems. They offer a rapid approach to create, deploy, and manage a too-rapidly growing storage arena. The concepts of abstract management hold great promise. But, when virtual definitions are deployed without careful engineering considerations of

underlying physical bandwidth and throughput requirements, application SLA targets can be woefully missed.

When scale-out file systems that use storage devices across each SAS Grid node are provisioned, the inherent physical problems of both direct attached and network attached storage come into play, along with a host of others.

Scale-out storage that is based on a virtual file system, placed across storage devices *internal to each node*, distributes data across the available drives in the shared cluster. This means that a file might be sitting on a virtual file system that spans multiple host nodes. If a node requests data that is physically stored on a device in another node, it must reach through the network interconnect to obtain it from that device. This involves the NIC cards on both nodes, the network fabric and switches, and the OS resources level of each host: CPU, memory, caching, and IO operations. Each of these resources takes time. That is quite a reach! Some scale-out software has built-in intelligence to load-balance and manage locality of reference for this storage paradigm, attempting to optimize file placement on a node that consumes it the most. Some offerings do not. Carefully plan your strategy with your vendor, know your data sharing and consumption patterns for performance, and test everything!! If you are dealing with smaller amounts of data, scale-out storage might be quite sufficient. However, it might present architectural performance challenges for very large volumes with heavy sharing and heavy IO.

In addition to data access performance, back-end data services that traditionally resided on SAN or NAS now come into play at the host-node level. This includes backing up, replication, compression, de-duplication, encryption, and so on. These will now be invasive to the VM host and host-level resources and should be taken into consideration when planning your scale-out file system. An example of this is erasure coding and replication levels in most scale-out file systems. Because data is written to more than one area for safety (a replication copy—remember, there is no SAN to do this for you now), writes across the network interconnect are very common and the data amplification adds to the workload. This places added stress on the host node resources as well as the fabric interconnect on which the storage is deployed. The SASWORK file system does not require this level of safety replication and, in some HCI arrangements, you cannot turn it off. Discuss this type of issue handling with your vSAN administrator.

Storage IO performance is crucial in scale-out storage arrangements in order to help ameliorate the penalty of the potential long reach for data. Choosing enterprise class flash storage is crucial for UNMAP support, efficient garbage collection, and so on, as the devices fill and have to work harder to find available space. See the following paper for ancillary reading:
http://support.sas.com/resources/papers/proceedings17/SAS0552-2017.pdf

LUN CONSTRUCTION SPECIFICS

There are multiple ways to create virtual storage file systems in vSphere, each possessing physical performance and management ramifications. This section will discuss common ways that virtual file systems are provisioned from an LUN perspective, pros and cons of each, and guidance of which to use in several situations.

VIRTUAL MACHINE FILE SYSTEMS (VMFS)

Most VM administrators use Virtual Machine File Systems to store their boot-up and configuration files, guest operating systems, software applications. They are often used for data storage. There are two primary types of VMFS file systems, Virtual Machine Disks (VMDK) and Raw Device Maps (RDM).

VIRTUAL MACHINE DISKS (VMDK)

VMware administrators prefer the utilization of VMFS/VMDK file systems. They are completely manageable with the VM console as SDS implementations and are completely supported by vSphere services (vMotion, Storage vMotion, VM Snapshots, and so on). There are multiple ways to implement VMDK file systems. Because VMDKs are abstract definitions, they can be moved around seamlessly from one place to another in the same storage arena, or to another storage array altogether. This is done with a software “click and manage” paradigm. Given their abstraction, they are perfect candidates for vSphere as mentioned above. One of the most common ways is to create the VMDK on a single storage LUN.

Single LUN

Creating a single storage LUN involves creating an LUN address on a storage system and mapping a VMDK to that LUN. This is the most common way to store the VM host's boot-up system, OS, application software, and so on. The VMDK can be moved easily for purposes of load balancing, tier placement, and so on by placing it on a new LUN address anywhere in the system.

The VMDK will use the capacity of a single LUN and is subject to the performance limitations of that LUN's underlying construction (capacity, how many devices it is striped across, RAID level, and queue depth setting on storage, for example). On large, all-flash arrays, or in the case of multiple flash drives striped together in an HCI arrangement in a host node, a single LUN can work relatively well. When a single VMDK is set up to serve as a data LUN for a SAS Grid file system, it requires enough physical bandwidth to deliver the minimum 125 MB/s per SAS CPU core metric that SAS Grid requires. As long as the underlying LUN is performant enough, it generally works well.

The other performance consideration of a single LUN is the queue depths of the LUN and the queue depths of the initiators and controllers that it is accessed from. Sometimes the queue depth is exceeded by the concurrent workload and becomes a crippling bottleneck. This can happen easily with large SAS Grid workloads, despite using maximum virtual and physical queue depth settings. It might become necessary to involve more LUNs for queue depth in order to avoid high service queues.

Multiple LUNs—Capacity Only or Series of Separate LUNs

If a single LUN does not service the capacity needs of a given file system, additional LUNs can be added to the VMDK. These LUNs are "capacity LUNs" and only provide additional buckets of space. They are not striped by vSphere with any other LUNs in that VMDK. As a result, they do not provide the benefit of parallel operations like striped physical LUNS do in a host logical volume. If this parallel performance benefit is required, the common method of creating a logical volume at the Guest OS level, composed of striped LUNs, is used. The top half of Figure 2 below illustrates the use of multiple "capacity" LUNs for VMFS.

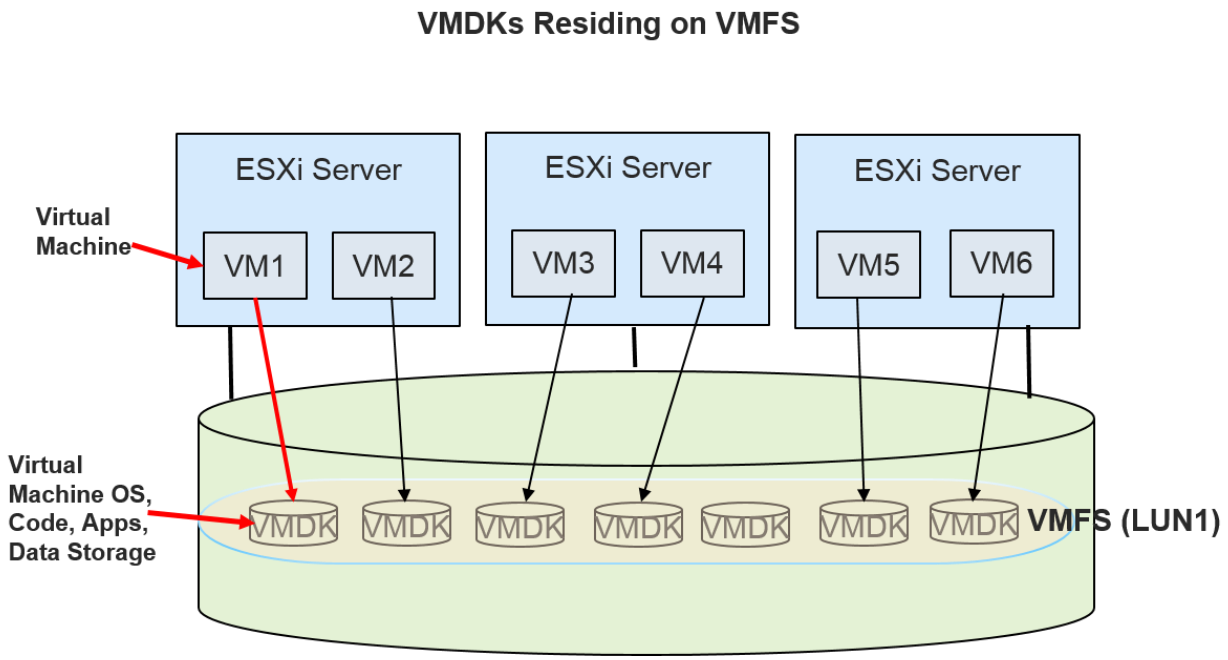

Figure 1. VMFS File Systems – Single LUN Placement

Multiple LUNs Striped By VM Host Guest OS

Creating multiple VMDKs under VMFS and striping across the LUNs of the VMDKs using an OS logical

volume provides better performance and higher queue depths than those of a single LUN. This creates parallelism but requires a file system mount through the host logical volume. The bottom half of Figure 2 below illustrates this concept. The same precepts as with bare-metal logical volumes typically apply here: 64 KB stripe size, 8 LUNs (VMDKs to stripe across) and an 8 MB (at least) Read Ahead. For best practices in bare-metal logical volume development in Red Hat RHEL 6 and 7, see the following: http://support.sas.com/resources/papers/proceedings11/342794_OptimizingSASonRHEL6and7.pdf

Before you proceed with any of these VMDK approaches, consult with your VM administrator to determine its impact on vSphere services and any additional management consequences.

Out of necessity, you have now moved beyond the VMFS and VMDK easy, software-defined management, ease of migration, growth, and so on. You are now in "hands-on management" territory. Many administrators do not like this. However, this is still a good trade, given that VM services like vMotion primarily benefit from managing "thin" resources, and SAS needs them to remain somewhat "thick." It is crucial to deploy storage in a performant manner, and the most reliable way to do that is to involve OS logical volumes striped over multiple LUNs.

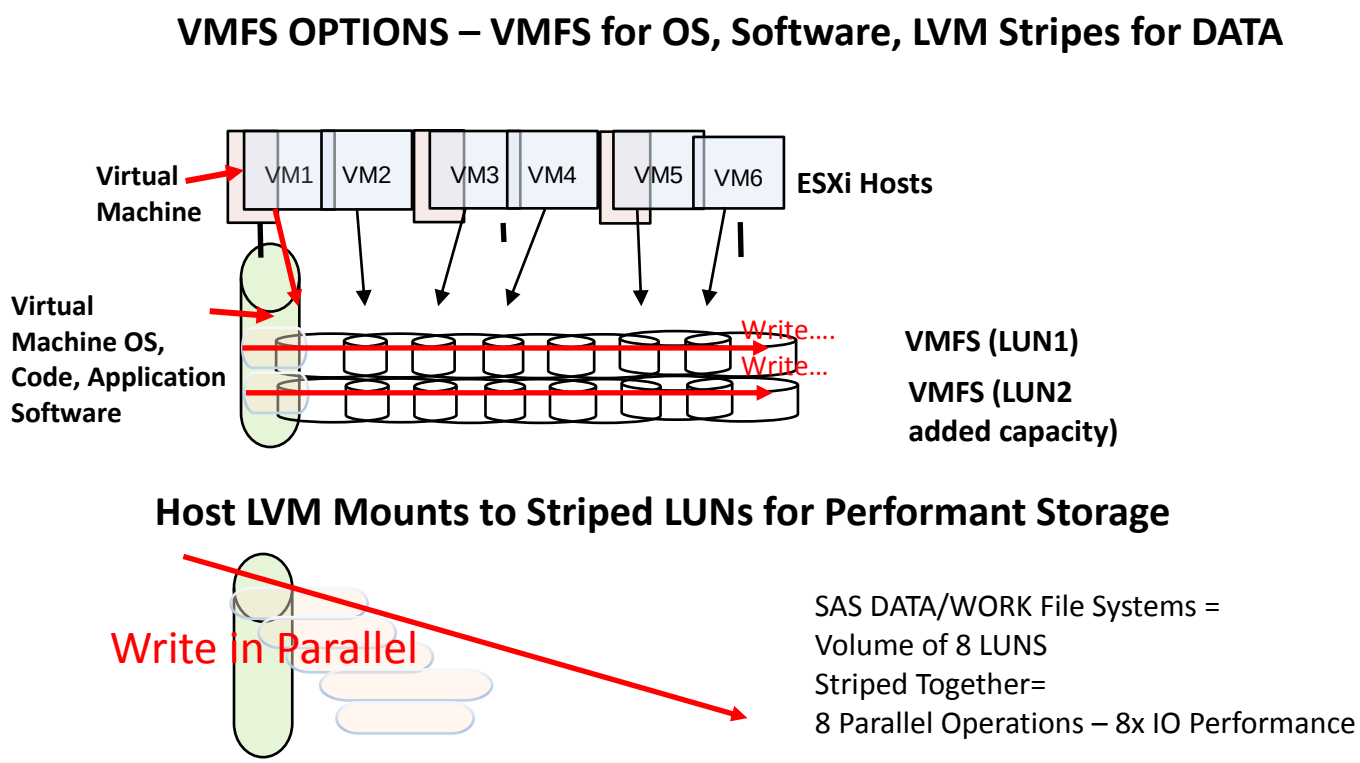

Figure 2. Multiple Single-LUN VMFS File Systems for OS, Software, and Code, Using Guest Host OS LVM Stripes Across Multiple Physical LUNs for IO Performance

RAW DEVICE MAP LUNS

At least one major shared file system vendor does not allow installation on VMFS/VMDK arrangements. IBM Spectrum Scale (formerly known as GPFS) requires installation via Raw Device Maps (RDMs).

Raw Device Maps are implemented as a mapping file that can be created in a VMFS volume and act as a pointer to, or a proxy for, a raw, physical device. RDMs are connected via an SCSI device, which is used directly by a virtual machine. The RDM contains metadata for managing and redirecting disk access to a defined physical device. In short, it is a pointer to physical back-end storage for the VM node (for example, a SAN LUN). It appears as a normal disk file to the VM node and has typical file systems

operations and services. Because the RDM is mounted via an SCSI device, each RDM requires an SCSI device port in the VM node. See Figure 3 below for an illustration.

Part of the mapping file includes metadata that defines the location of the physical device mount (for example, LUN address, physical disk volume, disk file, and so on). The metadata also tracks the state of the mapped physical device, permissions levels, whether it's locked by a VM, and so on.

With VMDKs available, why would a file system like Spectrum Scale require RDMs?

With a VMFS VMDK, you have vSphere services such as vMotion, Storage vMotion, and VM Snapshot. These services provide VMDK migration, backup, cloning, and so on, via the VM hypervisor resources on the VM network. These services are handled within the vSphere environment, taking full advantage of the virtual environment.

User communities report similar throughput performance for VMFS and RDM-enabled systems. In some cases, there are advantages or requirements to have RDM direct access to a physical device pool. Some high-performance databases and cluster file systems (for example, IBM Spectrum Scale clustered file system) require RDM access versus VMFS VMDK access. In general, the RDM allows storage access through a defined VMFS, but it keeps the performance advantages of array-level back-end services, LUN definition, and growth. This is especially helpful with large SAN volumes (for example, ~300 TB or greater). For large storage entities, using RDM can avoid the physical to PZV volume conversion to VMDK at the hypervisor layer, as well.

The RDM allows you some of the advantages of direct access to a physical device while keeping some advantages of a virtual disk in VMFS. As a result, it merges VMFS manageability with raw device access strengths. For example, it places a storage system that benefits from array-level scalable backup on the virtual machine system, and it uses a clustering scenario that spans physical hosts. These data movement operations, especially at large scales, can sometimes perform better via physical back-end services. With an RDM, you can still use vMotion to migrate virtual machines using raw LUNs, add raw LUNs to the VM using vSphere, and use third-party clustered file system features such as distributed file locking, naming, and permissions management.

There are two primary implementations of RDM: virtual (**RDMv**) and physical (**RDMp**). With RDMv, you retain the ability to use vMotion, Storage vMotion, and VM Snapshot services. With RDMp, you still retain the ability to use vMotion and Storage vMotion, but you lose the VM Snapshot services capability. Figure 3 shows that Raw Device Maps are just LUN addresses mapped to your VMFS volume. They represent back-end storage not completely managed by the VMware vSphere Hypervisor.

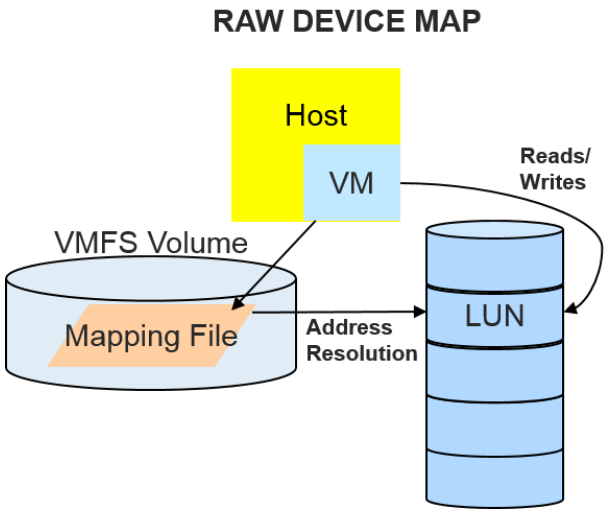

Figure 3. Raw Device Maps Are LUN Addresses Mapped to Your VMFS Volume

WHERE LUNS AREN'T USED

There are situations where LUN construction is not part of a storage architecture. Some HCI vendors do not require LUN addressing to spread a workload. An example is the Nutanix hyper-converged infrastructure. Nutanix presents containers to its own hypervisor. These containers are thinly provisioned and each can access all of the space in the clustered storage pool. As a result, free space is only managed at the storage pool layer and not within each container. So, the concept of not using LUNs can appear with some proprietary storage paradigms such as hyper-converged and scale out infrastructures. Careful testing with these types of infrastructures must be performed with your SAS Grid on VMware workloads. Acceding control to a proprietary storage management system warrants appropriate virtual and physical infrastructure engineering and performance planning.

CONCLUSION

SAS Grid can run very successfully on VMware ESXi vSphere hosts and their associated storage definitions. In order to achieve optimal SAS application performance on SAS Grid, both virtual and physical architectures must be engineered and tuned for performance. Basic decisions must be made about what will be managed by the virtual hypervisor versus the VM Guest OS in terms of BIOS settings, power, sleep settings, and so on. The virtual machine must reside in a performant manner on the physical hardware in terms of thick CPU and memory allocations, socket placement, and aligning full path bandwidth. In addition, queue depth settings must be adequate throughout the full virtual path: from back-end host initiators through physical LUN placement on storage physical host.

Choice of storage architectures and attachment is perhaps the most impactful decision. Whether choosing single LUN VMDK file systems on all-flash storage or managing host LVM stripes of LUNs to garnish bandwidth where needed, the underlying storage queue depth and bandwidth determine whether a SAS Grid compute node will perform well.

It is important to follow the ancillary guides listed in this paper for IO bandwidth planning, VMware provisioning and tuning, Guest OS tuning, and managing which services will be hypervisor-based versus relegated to back-end storage.

REFERENCES

Brown, Tony, and Margaret Crevar. 2015. “Frequently Asked Questions Regarding Storage Configurations.” Cary, NC: SAS Institute Inc. Available at: http://support.sas.com/resources/papers/proceedings15/SAS1500-2015.pdf.

Curran, Christopher. 2010. “Enabling Intel VT and AMD-V virtualization hardware extensions in BIOS.” In *Fedora 13: Virtualization Guide*. Red Hat, Inc. Available https://docs-old.fedoraproject.org/en-US/Fedora/13/html/Virtualization_Guide/sect-Virtualization-Troubleshooting-Enabling_Intel_VT_and_AMD_V_virtualization_hardware_extensions_in_BIOS.html.

Denneman, Frank, and Niels Hagoort. 2017. *VMware vSphere 6.5 Host Resources Deep Dive*. CreateSpace Independent Publishing Platform. ISBN 9781540873064. Available http://pages.rubrik.com/host-resources-deep-dive_request.html?utm_campaign=Rubrik-Blog.

Marson, Barry. 2017. “Optimizing SAS® on Red Hat Enterprise Linux 6 and 7.” *Proceedings of the SAS Global Forum 2017 Conference*. Cary, NC: SAS Institute Inc. Available http://support.sas.com/resources/papers/proceedings11/342794_OptimizingSASonRHEL6and7.pdf.

SAS Institute Inc. 2015. SAS technical paper. “Moving SAS® Applications from a Physical to a Virtual VMware Environment.” Cary, NC: SAS Institute Inc. Available http://support.sas.com/resources/papers/MovingVirtualVMware.pdf.

ACKNOWLEDGMENTS

The author wishes to acknowledge Lucas Figgs, Steven Bonuchi, and Joe Muniz from Dell EMC Corp.; Barry Marson from Red Hat, and Brian Porter from IBM for their contributed best practices for VMware vSphere implementations of their storage and shared file system products.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Tony Brown
SAS Institute Inc.
tony.brown@sas.com