PERFECTDISK
SMARTER OPTIMIZATION

Whitepaper

Getting More VMware Performance from Fewer I/O Requests

Taking Advantage of VMkernel Settings

Getting More VMware Performance from Fewer I/O Requests

Cormac Hogan, a well-respected and knowledgeable senior technical marketing architect with VMware, recently published an illuminating blog on the trade-offs VMware makes between performance and fairness[1]. The basic premise is that when there are multiple VMs on the same LUN, there needs to be some kind of arbitration in place to make sure that one busy VM does not consume the bulk of the resources at the expense of other, less busy machines. Cormac's blog identifies several settings internal to the VMkernel and their role in the performance versus fairness contest. His article also comes with a warning that these settings are pre-configured to allow the virtual machines to perform optimally and tampering with them may produce an undesirable outcome. Further, all of these settings are global and are not implemented on a per-LUN basis; so it would be wise to leave these settings alone.

The Root Cause of VMware Performance Issues

Most VMware performance issues are in the form of disk latency and I/O contention. Disk latency problems develop when I/O response time degrades to an unacceptable level. VMware rightly contends that if disk response time is greater than 30ms, you have a problem. Latency and contention problems arise when the storage controller is overwhelmed with I/O. As a result, the completion time for physical disk accesses gets longer and longer. Managing latency has very little to do with VMware and a lot to do with your guest OS. This will be discussed in more detail later. What the VMkernel settings are trying to do is strike a balance between performance and fairness across all virtual machines using a set of pre-defined metrics. There are things you can do to take advantage of these settings to ensure optimal performance for all your virtual machines in spite of VMware reducing the number of outstanding I/O requests per VM.

The settings described here are intended to mitigate situations that may arise from the work being done by your guest systems. Windows exhibits behaviors that create I/O contention and disk latency issues. Before we look at what you can do to improve virtual machine performance, let's examine these settings so we can understand the dynamics between the Windows operating system in the guest and the VMware host.

[1] http://blogs.vmware.com/vsphere/2013/04/virtual-machine-io-fairness-versus-performance.html

Disk.SchedNumReqOutstanding

This setting throttles queue lengths when more than one virtual machine is doing I/O to the same LUN. The default value for the setting is 32, meaning a virtual machine can issue up to 32 I/O requests. It is not uncommon to see a single VM have a queue depth of 64, but once a second VM starts to do I/O to the same LUN, the number is cut in half to ensure fairness. Queue depths may be trimmed with the addition of more active VMs on the LUN, as we see in the next setting.

Disk.SchedQControlVMSwitches

While *Disk.ScedNumReqOutstanding* is the maximum number of I/O for a virtual machine, *Disk.SchedQControlVMSwitches* determines if the number of I/O sent by a single virtual machine needs to be throttled down. This setting is the number of times there is a control switch between virtual machines. The default setting is 6 and during periods of high I/O, if we have not gotten back to service the outstanding I/O of any of the previous machines, the number of I/O a virtual machine can schedule will be throttled back. In the interest of fairness, you could be throttled down to 16 I/O requests.

Disk.SchedQControlSeqReqs

This setting is the inverse of the one above. If the number of I/O requests can be throttled down, there needs to be a way to throttle them back up. The default value is 128 scheduled I/O, which means a virtual machine can be scheduled 4 times without interruption (4 x 32). When this happens, *Disk.SchedNumReqOutstanding* returns to its default (32).

Disk.SchedQuantum

There may be occasions when there are multiple virtual machines sharing the same LUN but we may want a VM to request more I/O than allowed by *Disk.SchedNumReqOutstanding.* One such instance would be when sequential I/O is occurring. The rationale here is that there is a performance hit if you need to seek back to the same location to complete the sequential I/O the next time the VM is serviced.

Disk.SchedQuantum represents the maximum number of consecutive sequential I/O from a virtual machine before VMware switches service to another virtual machine; its default value is 8. If a sequential pattern is detected, *Disk.SchedReqNumOutstanding* may let the 32 I/O complete and then allow another 8 I/O as set by *Disk.SchedNumQuantum*. This will help the performance of all the virtual machines.

At this point, we need to define what constitutes sequential I/O. Sequential in this case does not mean contiguous or adjacent. Sequential in this instance means proximal. To determine what is proximal we need another setting.

Disk.SectorMaxDiff

The idea behind this setting is to determine if any sequential I/O are being performed when the concept of sequential means it is proximal. If the next I/O is proximal, the virtual machine gets the advantage of getting another 8 I/O requests, which means it should be served faster by the storage. If the next I/O is not proximal we will move on and service the next virtual machine. The default value *for Disk.SectorMaxDiff Diff* is 2000 disk sectors. In other words, if the previous I/O was within 2,000 disk sectors of the next I/O, there is a sequential pattern.

As Cormac explained, all of this is done to make sure all of the virtual machines are treated fairly in terms of having access to I/O resources on the host. So, if VMware is invisibly throttling I/O requests for your VMs and allowing extra I/O requests for virtual machines doing sequential I/O, what can you do to make the most of the situation?

Understanding NTFS

Most of the performance issues with VMware tend to be I/O-related. Both I/O contention and disk latency are the result of too many I/O and not enough resources. We have to remember that these I/O originate in the Windows guest. As an example, let's examine what happens when you create a 2GB file. Windows creates a record in the volume index and it asks the file system (NTFS) for some space, where it logically stores the file. Now, the goal is to provide this logical space as fast as possible, not to provide a best fit. For our example, and just to keep the math simple, we will assume the file system provides space in 1,000 locations in equal size 2MB chunks. It is important to remember all this is happening before anything is written to the disk. As noted, when there are multiple VMs connected to the same LUN, the default number of I/O requests is 32. For our 2GB file, this means the guest system passes 1,000 I/O requests of 2MB each across the hypervisor to the controller. To accommodate the whole file, the guest will go through its entire 32 I/O request stack a staggering 31.25 times (32 x 32.25=1000).

The storage controller has to process 1,000 SCSI commands of 2MB each and write them to the SAN. The absolute minimum number of physical disk accesses is 1,000 assuming each SCSI command is completed in one access. It is more likely the file will require 2,000-3,000 physical disk accesses due to the controller software splitting the 2MB requests across multiple disks.

There are several bottlenecks here:

- The first is **the total number of SCSI commands**. Excess SCSI commands means VMware is using more CPU and memory than necessary;
- secondly, **the VM can only issue 32 requests at a time**, so it takes just over 31 times through its queue to accommodate the whole file;
- **the controller has to process the same 1,000 SCSI commands** to find space for the file and finally;
- **there are in excess of 1,000 physical disk I/O needed** to write this out to the disk.

Now, consider that this situation is taking place every day, on every host, in every VM on most of the files in your organization. Every subsequent file access will use the same number of I/O unless the file is cached. Do you begin to see why contention, disk latency and poor I/O performance are issues?

Optimize the OS

There is nothing in VMware (or other virtualization platforms) or in the storage hardware that mitigates the problem. This problem starts with the behavior of the NTFS file system and that's where it needs to be fixed. Let's look at the same file creation scenario, only this time the user will have run Raxco's PerfectDisk drive optimization software beforehand. We create the same 2GB file, but this time the PerfectDisk OptiWrite technology intervenes. Instead of coming up with space in 1,000 locations, OptiWrite finds a contiguous 2GB chunk of space. The VM, limited to 32 I/O requests, makes a single request that is 2GB in size, using only one of its 32 I/O requests and freeing up the rest. This time our 2GB file is conveyed across the hypervisor in one (1) big SCSI command, so VMware uses less CPU and memory. The controller only has to process a single SCSI command for 2GB, which it can map to the storage in > 50 physical disk accesses (an arbitrary number but most certainly less than 100).

The use of PerfectDisk resulted in the following:

- The file was not logically fragmented by NTFS
- The larger file request maximized utilization of the available I/O requests (2GB versus 2MB)
- Fewer SCSI commands were needed to request the file eliminating hypervisor and controller workload (1,000 versus 1)
- Fewer physical disk accesses are needed to write the file > 1,000 versus < 50)

Contiguous files in the Windows guest dramatically reduce the total I/O workload.

Disk Optimization and VMware

Two years ago, VMware asked Raxco to quantify the benefits of disk optimization in a VMware environment. We set up a test using two sets of identical disks. One set was the baseline and the other was optimized with PerfectDisk. We ran the same benchmark on both sets and used VMware's vscsiStats utility to capture the results. The metrics showed the PerfectDisk disks produced a:

- 28% reduction in total I/O across the storage stack
- 1,200% increase in the number of large I/O (>524K)
- 49% reduction in the number of slow I/O (>15ms)
- 58% increase in sequential I/O (separated by 1 disk sector)
- 28% improvement in throughput

This is exactly what you would expect to see. By optimizing files, PerfectDisk reduced the total number of SCSI commands to the hypervisor and controller. The 2GB sample file previously in 1,000 fragments is now accessed with one SCSI command. When there are fewer I/O for the same amount of data, then each I/O is going to be bigger. Fewer and bigger I/O alleviates queue congestion and produces less work for the controller and physical disks. This means disk I/O are faster and latency improves. When files are contiguous, you get more sequential I/O as well. All of these metrics combine to provide better virtual system response, eliminate latency and contention and improve system throughput. The figures below illustrate the difference guest optimization makes.

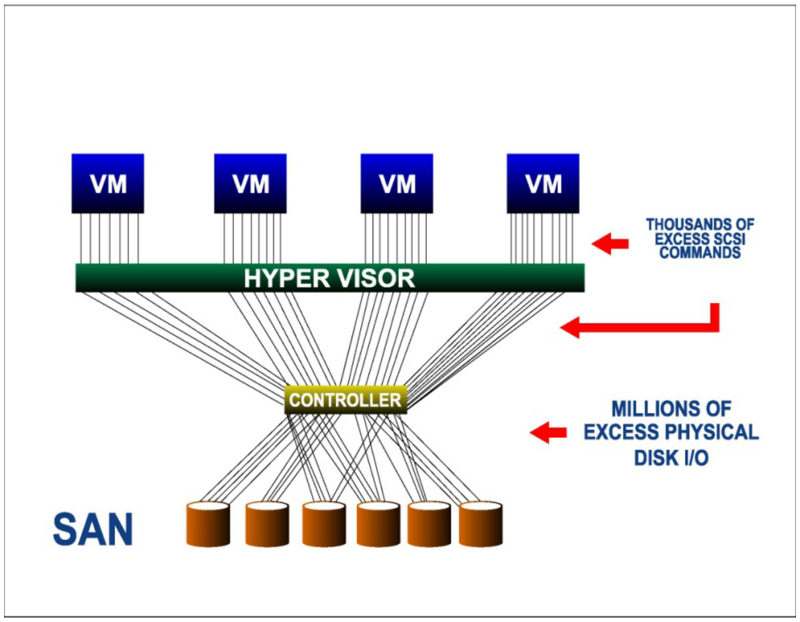
VM
VM
VM
VM
THOUSANDS OF EXCESS SCSI COMMANDS
HYPER VISOR
CONTROLLER
MILLIONS OF EXCESS PHYSICAL DISK I/O
SAN

Figure 1—Unoptimized: Lots of SCSI Commands and Disk I/O

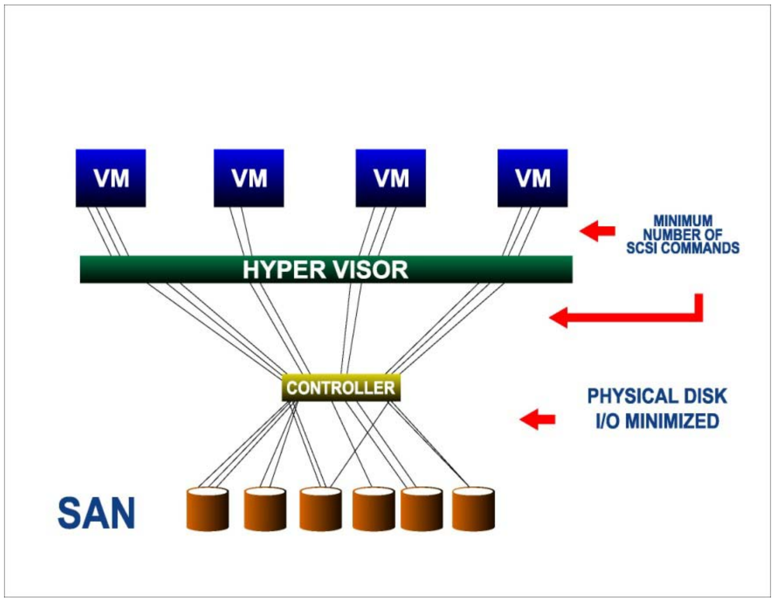

Figure 2—Optimized: Reduced SCSI Commands and Disk I/O

Now, let's tie these improvements back to the VMkernel settings. The *Disk.SchedNumReqOutstanding* setting is the maximum number of I/O requests a VM can have when more than one virtual machine is accessing the same LUN and its default is 32. In our example, the fragmented 2GB file needed 1,000 I/O requests to access the file. With the optimized file, only one I/O request was needed. Simply put, you get better throughput with one large I/O request than with 1,000 smaller ones. If the number of I/O requests is going to be throttled, make the most of the ones you have by generating larger I/O.

The *Disk.SchedQuantum* and *Disk.SectorMaxDiff* settings combine to give you up to 8 extra I/O requests when you are doing sequential I/O. In our example, the 2GB file comes to the controller on one chunk. The controller breaks this into smaller I/O across the disks in the SAN. It is far more likely that the controller will break a large I/O into smaller I/O that can be accommodated sequentially. This is what we saw in the VMware testing, where sequential I/O improved by 58%.

PerfectDisk generates sequential I/O in three ways:

1. The first is **making the file contiguous** so it moves in one big chunk.
2. The second is **free space consolidation**. Fragmented free space is the source of file fragmentation. When the file system can find contiguous free space, it is less likely to fragment any files.
3. Thirdly, the **OptiWrite fragmentation prevention** driver will eliminate up to 99% of any new file fragmentation once the free space is consolidated.

Benefits

There are several benefits to be derived from optimizing the Windows guest systems:

- Larger I/O gets more throughput when I/O requests are throttled by VMware *(Disk.SchedNumReqOutstanding)*
- Improved sequential I/O is favored with additional I/O requests *(Disk.SchedQuantum and Disk.SectorMaxDiff)*
- Fewer total I/O reduces the hypervisor and controller workload
- Fewer and larger I/O reduces HBA-LUN queue congestion
- Fewer and larger I/O improves controller mapping and reduces latency
- Consolidated free space and OptiWrite prevent up to 99% of new file fragmentation so disks stay in good shape much longer

These benefits provide optimum performance as the disk volumes fill, maximizing disk space utilization and storage life.

Summary

VMware does of good job of managing the things over which it has control. Through its VMkernel settings, it is able to ensure that in a multi-virtual machine environment, there are equitable resources available for all the virtual machines accessing the same LUN. Without these settings, the I/O-intensive VMs would tilt the resource equation in their favor at the expense of less busy machines.

What VMware cannot control is the onslaught of unnecessary I/O being sent by all of the VMs on the host. The Windows guest determines if a file is logically addressed in 1,000 locations or one. As an administrator, you can optimize the guest VMs to minimize this total I/O workload. Optimization will eliminate contention, improve latency, reduce queue congestion and increase sequential I/O. Disk optimization provides superior performance from all your VMs even while fairness is being imposed through the VMkernel settings. Efficiency is defined as more work per unit of time and that saves money.

Learn more about PerfectDisk in a VMware environment.

RAXCO SOFTWARE