VCP Tips Corner

Know Your VMs

Part1

www.virtualizeplanet.com

Author: Ricky El-Qasem: 2009

Ok again some elements relating VMs won't be covered in this document as I believe they are more relevant in other subjects. In this document I want to focus in on what makes up a Virtual Machine running in a vSphere environment so probably best place to start is the configuration of a VM i.e. how much memory a VM can see or how many vCPUs.

VM Item	*Value*	*VM Item*	*Value*
Memory	255GB	vCPUs per host	512
vCPU	8	VMs per host	320
vmkernal swap size (same as VM memory)	255GB	vCPUs per pCPU core	20
vmkernal swap files per VM	1	VMs per host in HA cluster with 8 or less hosts	100
SCSI Adapters	4	VMs per host in HA cluster with 9 or more hosts	40
SCSI device per adatpter	15	VMs per DRS Cluster	1280
vdisk file size	2TB - 512bytes	VMs per host in a DRS cluster	256
IDE devices per VM	4	VMs per 32bit VC -powered on	2000
Floppy Controllers	1	VMs per 32bit VC -powered off	3000
Floppy drives	2	VMs per 64bit VC -powered on	3000
NICs	10	VMs per 64bit VC -powered off	4000
Parallel Ports	3		
Serial Ports	4		
VMDirectPath PCI/PCIe devices	2		
VMDirectPath SCSI devices	60		

Reference:
http://www.vmware.com/pdf/vsphere4/r40/vsp_40_config_max.pdf

Memory:

The configuration matrix listed before really gives you a good grounding to what is a possible with virtual machine configuration so I'm going to spend time now to focus on features rather than parameters. VMware ESX unlike other hypervisors has some cool features for optimising that amount of memory needed for virtualisation. The following features are generalised as ***"over committing memory"*** . This refers to the over allocation of actual memory available to powered running virtual machines. The actual nitty gritty is facilitated by 3 internal features:

- **VMMEMCTL – the balloon driver**
- **The VMkernel swap file**
- **Transparent Page Sharing**

Whether you agree or disagree with ***"over committing memory"*** the real world evidence suggests, on the whole when resources are allocated to servers, on average only 15% of memory is utilised. So with ***"over committing memory"*** we can divert the wasted 85% to other system that way need it and thus reducing that amount of memory required by virtualisation.

The VMkernel swap file: When a VM is created a swap external to the VM is created with a file size as the same of memory which was allocated to the VM on creation. So if you specified that your VM should have 1GB then a file with the extension .vswp is created with a file size of 1GB. This swap file is used to dynamically re-enforce the amount of RAM a VM sees with page file instead. The VM is oblivious that some of the RAM it see is actually virtual memory in the form of a page file that resides external to the VM. To help you understand this further consider this example:

You have 1GB of physical RAM available for VMs and you have 2 VMs both configured to see 1GB of RAM. In theory they should only be able to see 512mb of RAM each but ESX is able to dynamically tip the balance of which VM get actual physical RAM as a percentage opposed to being fooled which virtual memory.

Again consider on average on 15% of memory is utilised.

There is a way to control a reserved amount of physical RAM that is a guaranteed that a VM will always see. To set this you can change the reservation value in the settings of a virtual machine.

It's also worth noting that this swap file is not the same one that is created within the VM. If you choose to use a OS swap file like the windows page-file this virtual memory is different to the one we refer to here and is created when the OS was installed.

Reference
http://www.vmware.com/pdf/vsphere4/r40/vsp_40_resource_mgmt.pdf#page=28

VMMEMCTL – the balloon driver: By default when the VMware tools are installed in a VM a memory driver is installed as well. This memory driver aka **vmmemctl** can be called upon by the Vmkernel (ESX) to prompt the guest OS to page out (on to disk) idle pages of memory so they can be **RECLAIMED** and diverted to another VM which maybe struggling for memory resources. This process is known as ***"ballooning"***. Excessive ballooning is BAD as it means a lot of paging is happening. Using a page-file as virtual memory in a VM is good when you are restricted by not having enough memory but causes a performance overhead. It's worth noting that by default only 65% of the unreserved memory is available for reclamation unless you change this value in the advanced settings of the ESX server.

Reference
http://www.vmware.com/pdf/vsphere4/r40/vsp_40_resource_mgmt.pdf#page=28

Transparent Page Sharing: Now this is a cool feature and poses the least impact on performance. With this feature ESX is able to optimise the memory consumed by similar VM operating systems by not duplicating pages of identical paged memory. Because of the way operating systems like windows consume specific pages of memory which is mirrored for every instance of the same type of operating system. ESX is able use a pointer to reference a single instance of the page between multiple VMs. This feature alone can save 30% of the memory that would have been required without it enabled. If a VM attempts to modify this reference page ESX will perform a copy on right and point the VM to a new copy of the page. This feature is often compared to the de-duplication concept used in storage.

Reference:
http://www.vmware.com/pdf/vsphere4/r40/vsp_40_resource_mgmt.pdf#page=32

CPU:

ESX is able to time-slice the execute time of a processor core so a single physical CPU core is shareable between multiple VMs. Each VM sees a virtual CPU (vCPU) and is oblivious to the fact that there may be other VMs also sharing this processor. We have to be careful when we use the term vCPU and not be under the illusion that CPU tasks issued by the guest operating system scheduler are processed on an emulated device, meaning that the CPU threads are not execute in software but rather are executed directly on the physical processor.

A single vCPU can use as much CPU time one core is able to provide, so for example if you have a 3GHz dual core CPU and your VM has 1 vCPU , the VM can only ever use at maximum 3GHz. Of course the VM can be configured with more than 1 vCPU. By default VMs will get an equal SHARE of the CPU resource but we can tip scales in favour of a VM by changing this share value. ESX aggregates the total CPU resources and makes it available for sharing across all running VMs. In a example where all VMs have equal share a VM with 2 vCPUs will have double the priority over a VM that has 1 single processor.

Reference:
http://www.vmware.com/pdf/vsphere4/r40/vsp_40_resource_mgmt.pdf#page=15

Just like memory resources it's also possible to restrict the amount of CPU resources that are available to a single VM. This parameter found in the VM settings is known as the Limit. If you have a VM with multiple vCPU the Limit value is shared between the vCPUs. In the same way you are able to define a reservation in CPU MHz so that the VM is always guaranteed X amount of CPU cycles.

Reference: http://www.vmware.com/pdf/vsphere4/r40/vsp_40_admin_guide.pdf#page=155

ESX will probe every 20ms to see if a VM is struggling for CPU resources and will dynamically migrate the VM on to a CPU core that has available resources. It is also possible to tie down a single vCPU to a specific CPU core using the affinity settings.

Reference: http://www.vmware.com/pdf/vsphere4/r40/vsp_40_resource_mgmt.pdf#page=20

NIC:

When you create a VM you can choose between 4 different types of virtual network cards vNICs and its probably a good idea to get familiar with the differences. The chooses include:

- Flexible
- E1000
- Enhanced vmxnet
- Vmxnet3

Flexible: This adapter is used with 32-bit guest operating systems. At first before the VMware tools are installed it emulates a AMD PC32Net NIC aka Vlance. This provides greater driver support as most operating systems have a built in driver for this card. When the VMware tools are installed this vNIC is transformed into a vmxnet NIC which a VMware priority software NIC and out performs the Vlance.

E1000: This NIC is a software emulation of a popular Intel NIC (Intel Pro 1000). It is the default option for 64bit machines.

Enhanced vmxnet: This is a even faster performing verion of the vmxnet NIC and supports jumbo frames.

vmxnet3: Again this is another upgrade on the vmxnet family an out performs the previous versions. There is a caveat though with this option, its only available with version 7 hardware and thus restricts the migration of the VM to ESX 3.x hosts. Vmxnet3 also supports jumbo frames and is needed is you want to make use of the Vmkernels TSO (TCP Segmentation Offload) feature.

Reference:
http://www.vmware.com/pdf/vsphere4/r40/vsp_40_admin_guide.pdf#page=162

It's worth noting that the vmxnet family of vNICs are not software emulations unlike the Vlance and the E1000 but rather a paravirtualised adapter. With paravirtualised NICs the network traffic is passed directly to the physical NIC rather than initially being processed by the software vNIC first.

Reference: http://www.vmware.com/pdf/Perf_Best_Practices_vSphere4.0.pdf#page=33

When I was a VMware instructor I used to use a trick question to catch my pupils out: Q If a vNIC is reporting in the guest operating system that its speed is 1Gb, how fast is it processing network traffic. Well usually I get all sorts of answers like 10Mb or 100Mb but the trick with this question is the answer: A. As fast as the underlying hardware is able to transport the data. So the reason for that question is if you have a scenario where a VM is communicating to another VM that resides in the same host on the same vSwitch there's a good chance that they are communicating way faster than 1Gb even though that this is the reported connection speed.

MAC Addresses:

Look out for this, the MAC addresses assigned to physical NICs are not important anymore whilst used in a VMware ESX host. MAC addresses are assigned to the vNICS and there is 2 types of assignment that are important for you to understand. There is Automatic assignment and Manual assignment.

With Automatic assignment a MAC address is automatically generated by ESX. ESX will use an algorithm to attempt to provide a unique MAC address for the vNIC. Alternatively you can manually specify a MAC address. VMware has its' own Organisationally Unique Identifier (OUI) in which MAC addresses will start with 00:50:56

Reference: http://www.vmware.com/pdf/vsphere4/r40/vsp_40_esx_server_config.pdf#page=56

Disclaimer:

Whilst this document is useful to consolidate the amount of revision needed for the VCP exam it's not intended to help you cheat your way through the exam. To find this information useful you will needed to have been on a certified VMware course at some point. Also the information in this document by itself isn't enough and you need to follow the reference links. The intention of this document is to kick start you and point you in the right direction.