Carlos Pereira (capereir)

Introduction

Intel VMDq Technology provides significant performance gains for Ethernet traffic within a Virtual Machine (VM). By simply enabling VMDq within VMware ESX 3.5 U1 and later network throughput can as much as double while reducing CPU utilization.

With fine tuning within ESX, the performance benefits of Intel VMDq can be increased even further. This document provides an overview of how to enable the Intel VMDq technology.

Enabling Intel VMDq

At this time, the only Ethernet controller supported by VMware ESX with Intel VMDq technology is the Intel 82598 10G Ethernet Controller. While there is in-box driver support for this Ethernet Controller with VMware ESX 3.5 U1 and later, the Intel VMDq hardware assist technology is not enabled by default. To enable the Intel VMDq technology under VMware ESX is a two step process, outlined below.

Enabling NetQueue Support within VMware ESX

The Intel VMDq technology requires the Hypervisor – in this case VMware ESX - to make use of it. VMware refers to this support as 'NetQueue'.

To enable NetQueue, use the VMware Infrastructure client to access the server, clicking on the configuration tab, then advanced settings. (**Configuration→Advanced Settings**)

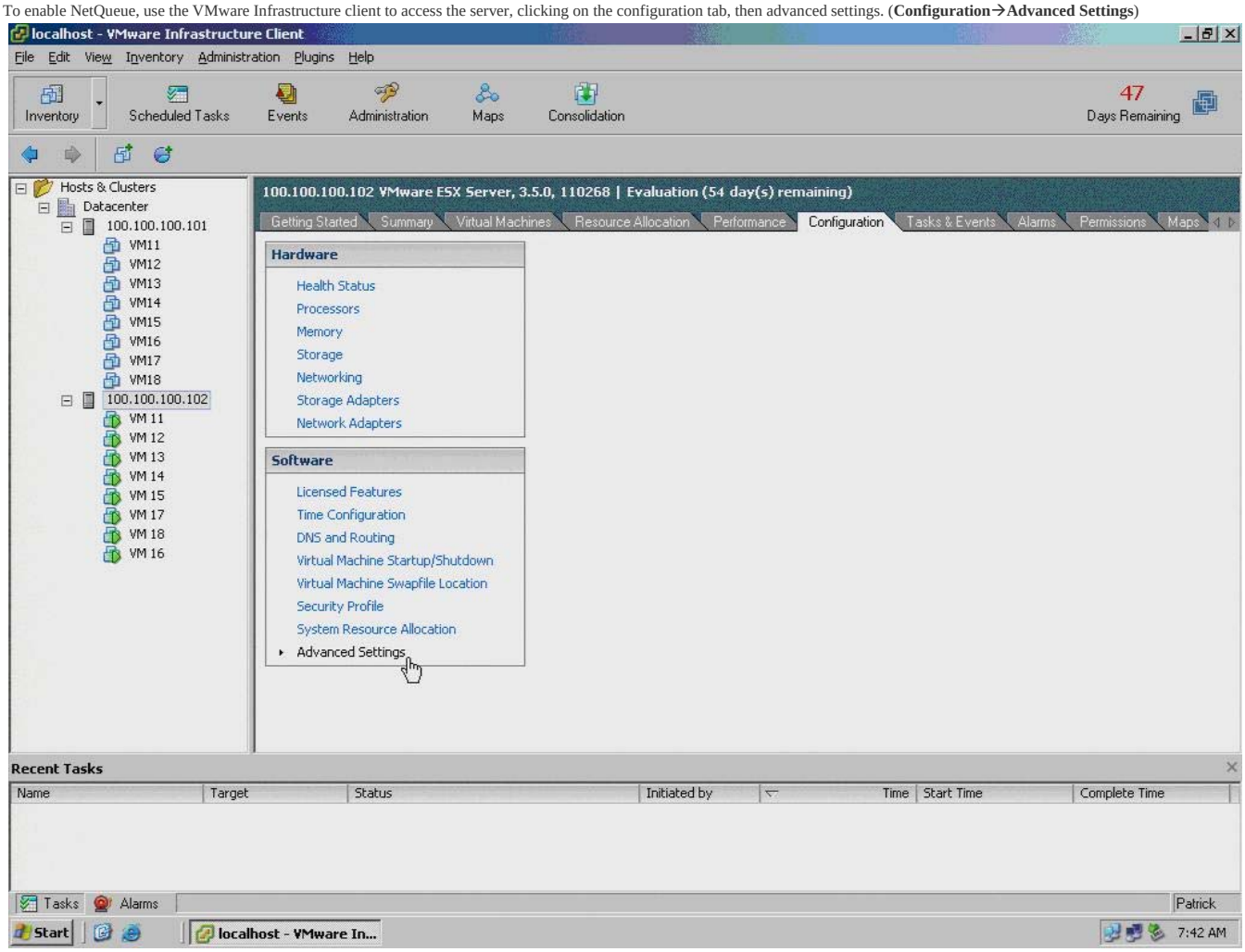

Figure 1 Opening the Advance Settings Page

Within the advanced settings tap, select VMkernel from the left tree control, then scroll down on the right side until the **VMkernel.Boot.netNetqueueEnabled** option is visible. Make sure this is checked. **(Configuration→Advanced Settings→VMkernel→ VMkernel.Boot.netNetqueueEnabled)**

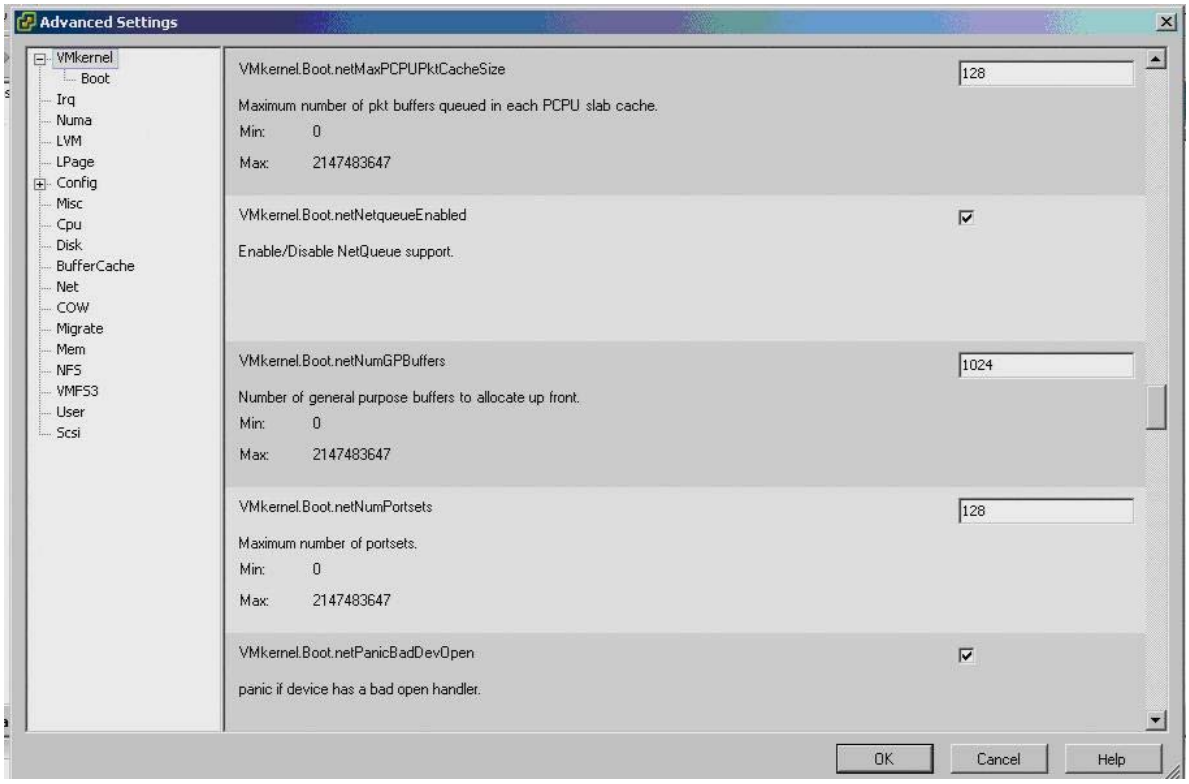

Figure 2 Advanced Setting - Enabling NetQueue

Configuring the Intel ixgbe driver to use VMDq

The Intel driver does not have the Intel VMDq technology enabled by default, it must be configured, and configured from the ESX console.

To set the required NetQueue options for the ixgbe module:

- MSI-X enables the Ethernet controller to direct interrupt messages to multiple processor cores. This feature is required for NetQueue to function with VMDq. In the examples below, the parameter `InterruptType` with a value of 2 specifies MSI-X.
- A value for `VMDQ` must exist to indicate the number of receive queues. A value of 16 for `VMDQ` sets the number of receive queues to the maximum. The Intel 82598 10 Gigabit Ethernet Controller provides 32 transmit queues and 64 receive queues per port, which can be mapped to a maximum of 16 processor cores. The range of values for the `VMDQ` parameter is 1 to 16.

For a single port and the maximum number of receive queues, use the command:

```
# esxcfg-module -s "InterruptType=2 VMDQ=16" ixgbe
```

For two ports, add the values in a comma-separated list for each parameter as shown in the following example:

```
# esxcfg-module -s "InterruptType=2,2 VMDQ=16,16" ixgbe
```

Reboot Required

After NeqQueue support has been enabled (**Configuration→Advanced Settings→VMkernel→ VMkernel.Boot.netNetqueueEnabled)** and the Intel ixgbe driver configured, a reboot of the system is required.

For further information regarding the basic enablement of the Intel VMDq technology with VMware ESX, see the following article: http://kb.vmware.com/selfservice/search.do?cmd=displayKC&docType=kc&externalId=1004278.

Virtual Machine Requirements

In order for VMDq and NetQueue to provide performance benefits, the VM should use the VMXnet Virtual Adaptor. This provides significant performance gains compared to the default Advanced AMD virtual adapter. Please refer to VMware documentation on how to ensure this Virtual Adaptor is installed and configured.

In addition to the VMXnet Virtual adaptor being installed, the VM will also VMware tools installed as well in order to achieve the performance benefits of Intel VMDq and VMware NetQueue technologies.

Optimizing VMDq Performance

After enabling VMware ESX NetQueue support and configuring the Intel ixgbe driver, the virtual machines will likely see improved network performance, while the system itself should see a reduction in CPU utilization.

Testing shows that on average, the maximum throughput to all aggregated VM's without the use of Intel VMDq and VMware ESX NetQueue technologies is around 3.5Gb/s. By enabling these

technologies, that throughput can be increased to around 7Gb/s.

For many, this is all that is necessary, the Intel VMDq technology provides hardware assisted sorting of incoming Ethernet packets. By offloading this activity from the host processor, throughput is increased and CPU utilization is reduced.

Intel testing has achieved near line-rate (9.2 Gb/s) performance under specific conditions. This section outlines how to fine-tune setting under VMware ESX to achieve optimal Ethernet performance.

Criteria for Optimal Performance

There are a number of factors involved with achieving the near line-rate performance.

- Number of CPU Cores
- Number of Virtual Machines
- Number of Queues
- Affinitization of VMs and Queues to Cores

Number of CPU Cores, Virtual Machines and Queues

To achieve the optimal performance, there should be no more than one VM per CPU core. Additionally there should be available at least 1 VMDq queue per virtual machine. This means that an 8 core system can use a single port Intel 82598 10 Gigabit Ethernet Controller to achieve optimal performance.

VMDq will still function and provide significant benefits if there are multiple VM's per CPU Core and VMDq queue – however at a reduced performance from what is described above.

Affinitization of VMs and Queues to Cores

When an Ethernet packet comes into the VMDq enabled Intel Ethernet Controller, it is examined and sorted into different Queue's based upon destination MAC address. Each queue is associated with a MSI-X interrupt – when a packet is placed into a queue, a MSI-X interrupt is fired.

A MSI-X interrupt will interrupt a specific CPU core – if the VM the incoming Ethernet packet is destined for is running on that core then VMware ESX will simply copy the data to the VM. If however the target VM is actually running on a different CPU core at that time, then that MSI-X interrupt must be handed off from the CPU core that received it to the correct core, resulting in a performance hit.

In general VMware ESX does a very good job at aligning VM's to cores and queues to cores. However it is not guaranteed to always be in alignment and there is a pretty significant performance cost to having one CPU core interrupt another to deal with incoming data packets. By ensuring that a specific CPU core is tied to a VM and a VMDq queue, the performance can be maximized.

There is a way to configure VMware ESX so that a VM is associated (affinitized) to a specific core and that a specific VMDq queue (and associated MSI-X interrupt) is affinitized to the same CPU core and VM.

Assigning CPU Core Affinity to a VM in ESX

Using VMware Infrastructure Client it is fairly simply to affinitize a VM to a specific CPU Core. Select a VM and right click, then select **Edit Settings→Resources→Advanced CPU** then select a CPU core (processor) affinity. For the purposes of this paper, the example has 8 VMs assigned in order.

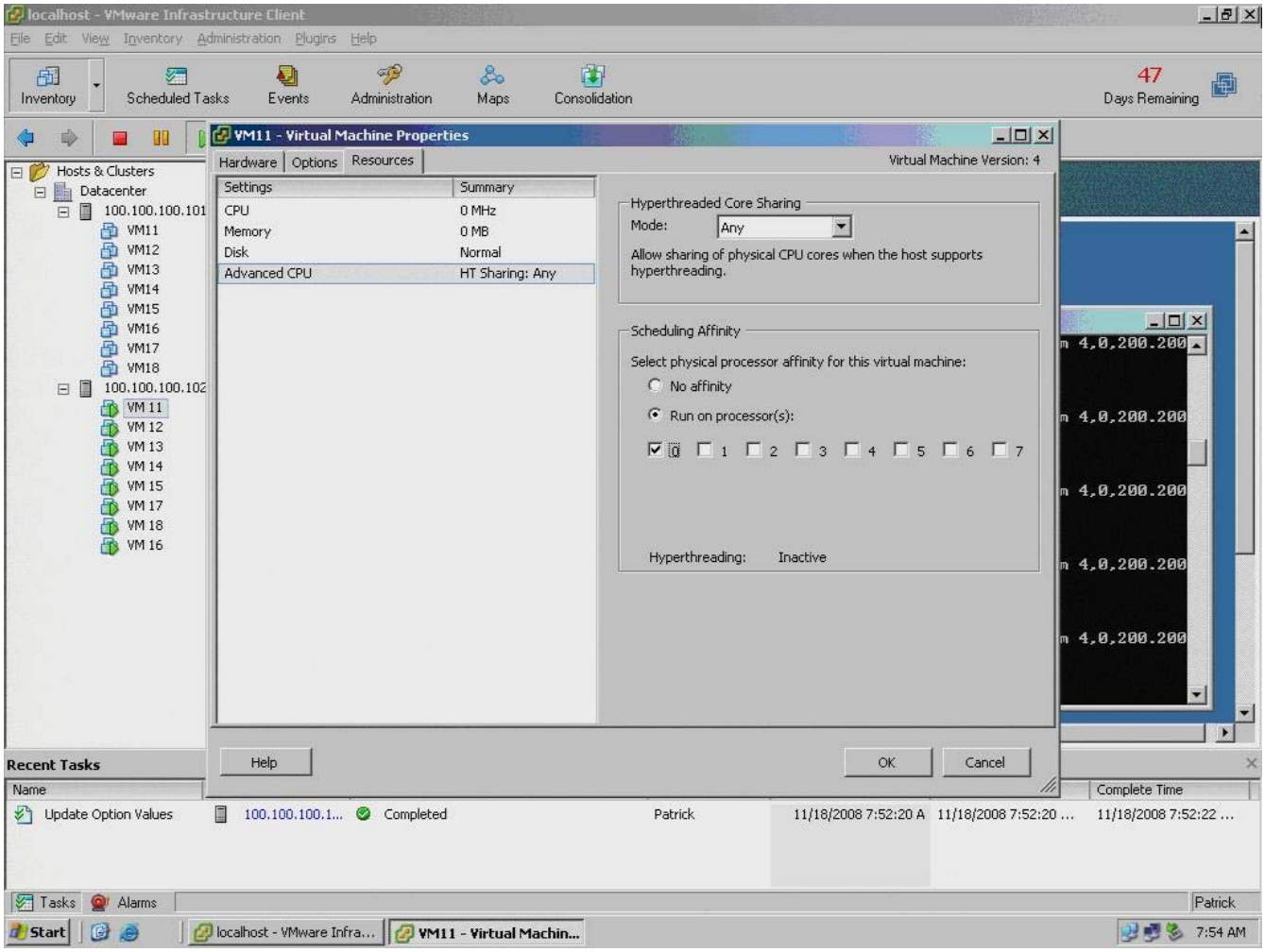

Figure 3 Affinitizing a VM to a CPU Core

Each VM in turn should be affinitized in this manner, with one VM per CPU core for optimal performance.

Ensuring Queue Association with a VM

In order to achieve optimal performance, the VM, the VMDq queue and associated MXI-X interrupt must be all tied to a core. VMDq queues are assigned when the Ethernet Adaptor from the VM comes online.

It is easiest if the VM are brought online in order, and the Core assignment is ordered the same way. The VM's can be started in order, making sure not to start one VM until the previous one has been fully loaded and is functional.

An alternate method is to disable the Virtual Ethernet Controller (VMXnet) in all of the VM's, and then bring them online in order.

Setting CPU Core Affinity for Intel VMDq Queues

This section describes how the CPU core affinity can be set for the MSI-X vectors.

This configuration must be done from the ESX console. The first step is to determine the interrupt vector associated with a given VM. It is recommended that the VM's be started in the same order as they are affinitized to CPU cores as described in section 3.2.

To do this, issue the following command:

```
# cat /proc/vmware/interrupts | grep vmnicN
```

Where `N` is the NIC number for the Intel Ethernet Controller with VMDq enabled. For this example, it is NIC 4.

```
[root@ESXVMDQ root]# cat /proc/vmware/interrupts | grep vmnic4
0x22:    51854877  111021889  128604809  130132571   28447187   29032332   27632
400   28885085 VMK vmnic4:v0-TxRx
0x2a:     7713326    6228250    6294704    6132591   43510540   45368675   37031
701   30891364 VMK vmnic4:v1-Rx
0x32:     7709465    6493047    6508735    6353188   43027123   44171780   30352
497   37375865 VMK vmnic4:v2-Rx
0x3a:     7917744    6535867    5780930    6195456   38475607   29038934   42771
131   44211374 VMK vmnic4:v3-Rx
0x42:     8260080    6360575    6222714    6141047   30031205   35550296   42842
518   41590096 VMK vmnic4:v4-Rx
0x4a:    46692431   36153788   40441573   28143260    7327923    7129655    7218
797    7386919 VMK vmnic4:v5-Rx
0x52:    48177588   37944617   27924431   39107305    6452898    6936479    6868
250    6907375 VMK vmnic4:v6-Rx
0x91:    36323321   36842632   41128155   41592524    7174432    7362735    8047
375    7614448 VMK vmnic4:v7-Rx
0x99:     7192338    6882219    8483287    8109056    4220630    3991409    4498
874    3834545 VMK vmnic4:v8-Rx
0xa1:      113333      93331     108375     104893      92820      86956      90
782      88485 VMK vmnic4:lsc
[root@ESXVMDQ root]#
```

Figure 4 Determining Interrupt Vector of VMs

In the example above, there are 9 interrupts associated with the VMDq queues. The 1st one is what is referred to as the default queue, it is used for transmitting data from all of the VMs and is where incoming Ethernet traffic not destined for one of the designated queue's is placed.

The first queue is associated with interrupt vector 0x2a, the second with 0x32 the third is 0x42 etc.

To affinitize a MSI-X interrupt associated with a specific VMDq queue, issue the following command:

```
# echo move <vector hex number> <pcpu> >/proc/vmware/intr-tracker
```

Where vector number comes from the steps described above and pcpu is the CPU core to affinitize the MSI-x vector with.

```
[root@ESXVMDQ /]# echo move 2a 0 > /proc/vmware/intr-tracker
[root@ESXVMDQ /]# echo move 32 1 > /proc/vmware/intr-tracker
[root@ESXVMDQ /]# echo move 3a 2 > /proc/vmware/intr-tracker
[root@ESXVMDQ /]# echo move 42 3 > /proc/vmware/intr-tracker
[root@ESXVMDQ /]# echo move 4a 4 > /proc/vmware/intr-tracker
[root@ESXVMDQ /]# echo move 52 5 > /proc/vmware/intr-tracker
[root@ESXVMDQ /]# echo move 91 6 > /proc/vmware/intr-tracker
[root@ESXVMDQ /]# echo move 99 7 > /proc/vmware/intr-tracker
[root@ESXVMDQ /]#
```

Figure 5 Affinitizing MSI-X interrupt with a CPU core

Using the example above, to assign the MSI-X interrupt vector from VMDq queue 1 with a core 0, where VM number 1 has been affinitized the following command is issued:

```
# echo move 2a 0 > /proc/vmware/intr-tracker
```

Figure 5 shows the steps taken to perform this action with all 8 VMs used for this example.

Drawbacks

Intel testing shows that performing the optimizations described within this document can achieve near line-rate throughput using the Intel 82598 10 Gigabit Ethernet Controller. This is achieved under specific conditions:

- One VM per core
- VMXnet Virtual Ethernet Adaptor and VMware VMtools installed in the VMs
- VMDq Queue and MSX-x interrupt is affinitized to VM and Core

If the number of VMs is increased beyond one VM per core, the benefits of the affinitization will begin to drop.

The optimizations and affinitizing discussed in this document have an additional side-effect. Implementing them will break the ability to utilize VMware's VMotion capability.

Summary

VMware ESX NetQueue technology and Intel VMDq provide significant Ethernet throughput gains as well as better CPU utilization by simply enabling the technologies as described in Section 2.

Under specific conditions and with some restrictions, these performance gains can be improved even further by ensuring that VMs, queue's (MSI-X interrupts) and CPU Cores are all properly affinitized.