

## table of contents

We recently spent some time troubleshooting some network performance issues with VMware ESX server running on our IBMx336 servers and thought that what we learned might be interesting to others. Network performance within a VM has always been only so-so, but generally good enough. With RHEL3/4 VMs on ESX 2.5 we were generally able to get 400-500Mb/s out of a VM using iperf, about half what the physical hardware could do, which I guess isn't too bad. With ESX 3 we had seen that number rise significantly, with RHEL4 VMs running the vmxnet driver approaching 800-900Mb/s within about 10% of the speed we were seeing on physical hardware.



Unfortunately, this performance increase didn't occur on all hardware platforms, specifically, our IBM HS20 and Dell 1850 servers saw a huge boost in network performance with ESX 3, while our IBMx336 system continue to perform at about 1/3 the speed of the physical hardware. This really showed up after we switched one of our remote sites from a fiber channel based SAN to an iSCSI solution.

When we first decided to switch from our fiber channel SAN (EMC Clariion CX series) to a simpler iSCSI solution (Equallogic PS series) at our remote sites we did so based on performance numbers that showed we could actually get better performance numbers out of the iSCSI solution, even with software iSCSI, that with the aging, and far more complex CX solution. When we brought the PS300E into the lab for testing against an HS20 blade we were amazed that we could achieve performance of 85-90MB/s between a single VM using a single GigE link, near 90% utilization. Running two VMs using two GigE connections got us to 180MB/s pretty easily, significantly faster than we had ever managed to get our CX400 to achieve.

We purchased our PS300E and installed it at the remote site where ESX server was running on IBMx336 systems. We then re-ran the same benchmarks against the PS300E using the production systems. This time the performance numbers were not so good, with transfer rates in the 30-40MB/s range. We brought the array back to the lab site and ran our reference benchmark against the new PS300E, performance was around 90MB/s, right about what we expected.

We really couldn't understand what was going on, the specs between the two hardware platforms were nearly identical, Dual Intel Xeon 3.2Ghz processors, 8GB RAM, Broadcom GigE NICs. Why would there be such a disparity? We suspected a network layer problem but after looking at switch configs over and over we just didn't see any of the usual suspects (duplex mismatch, increasing error counts, etc).

Since we were moving from Fiber Channel to iSCSI we had purchased a Intel Pro/1000 GT Quad card to replace the Qlogic FC HBA (yes, we could have purchased a hardware iSCSI HBA but our testing had indicated that this really would not be required to achieve the level of performance we needed and, based on forum feedback, the VMware support for Qlogic iSCSI adapters seemed a little flaky). We installed the Intel Pro/1000 into one of the IBMx336 cards and re-ran the test. Boom, performance was back in the 90MB/sec range.

We could have probably stopped there, we didn't absolutely need to use the onboard ports, however, now we were curious why the Broadcom onboard ports on a x336 would perform so much worse than the onboard Broadcom ports on an HS20. We began to investigate possible scenarios and ran across something interesting, when looking at the various files in /proc we came across the /proc/vmware/interrupts file and noticed it looked like this (edited slightly to fit on the page):

```
Vector PCPU 0      PCPU 1      PCPU 2      PCPU 3
0x21:    2        0        0        0 COS irq 1 (ISA edge), <VMK device>
0x29:    0        0        0        0 <COS irq 4 (ISA edge)>
0x31:    0        0        0        0 <COS irq 6 (ISA edge)>
0x39:    0        0        0        0 <COS irq 7 (ISA edge)>
0x41:    0        0        0        0 <COS irq 8 (ISA edge)>
0x49:    0        0        0        0 <COS irq 9 (ISA edge)>
0x51:   15        0        0        0 COS irq 12 (ISA edge)
0x59:    0        0        0        0 <COS irq 13 (ISA edge)>
0x61:    1        0        0        0 COS irq 14 (ISA edge)
0x69:    0        0        0        0 <COS irq 15 (ISA edge)>
0x71: 68505248    0        0        0 COS irq 16 (PCI level), VMK vmnic6,
VMK vmnic1
0x79:    108      0        0        0 COS irq 18 (PCI level)
0x81:    0        0        0        0 COS irq 19 (PCI level)
0x91: 155710     649201    289501    732347 <COS irq 20 (PCI level)>, VMK ips
0x99: 21577461   60183980   21031867   51667695 <COS irq 21 (PCI level)>, VMK vmnic2
0xa1: 106442     348866     139914     365738 <COS irq 22 (PCI level)>, VMK vmnic3
0xa9: 9503152    27310088    9603280    24230277 <COS irq 23 (PCI level)>, VMK vmnic4
0xb1: 106439     348786     140019     365653 <COS irq 24 (PCI level)>, VMK vmnic5
0xdf: 417335059 417060518 418055868 416873770 VMK timer
0xe1: 3739008    1903562    4602326    2230321 VMK monitor
0xe9: 92395997   68149255   95110679   69560265 VMK resched
0xf1: 15628      55045      42367      53573 VMK tlb
0xf9: 214270     0          0          0 VMK noop
0xfc: 0          0          0          0 VMK thermal
0xfd: 0          0          0          0 VMK lint1
0xfe: 0          0          0          0 VMK error
0xff: 0          0          0          0 VMK spurious
```

Of note is Vector 0x71 which shows two things, first, notice that interrupts for this vector are not distributed evenly to all of the CPUs. Looking further to the right you can see that this is because the Console OS (COS) has at least one driver with ownership of this interrupt (note the lack of angle brackets around the "COS irq 16 (PCI level)" in the far right column), while the VMkernel (VMK) has ownership of both vmnic1 and vmnic6, which matches up with the two Broadcom onboard adapters. Since the Console OS owns runs only on the first

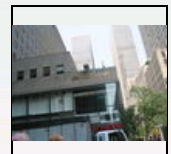


August '10							Su
Mo	Tu	We	Th	Fr	Sa		
2	3	4	5	6	7	8	1
9	10	11	12	13	14	15	8
16	17	18	19	20	21	22	15
23	24	25	26	27	28	29	22
30	31						29





Jesse Sighting's Weblog



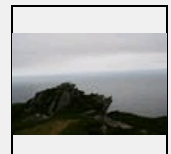
Me & Britt - NYC - Day 6



Me & Britt - Washington DC - Day 1



Rod & Tom in Ireland - Summer 2006 - Week 2



Rod & Tom in Ireland - Summer 2006 - Week 2

Visit My Photo Gallery



- ☐ Personal
- ☐ Football
- ☐ Technology
- ☐ Linux
- ☐ RHEL

Go!

All categories



serendipity

processor this forces all interrupts to be handled by the first processor, and then this processor must execute the interrupt handler for the Console OS and then the VMkernel which requires context switches and a fair amount of overhead. This is actually fairly well documented for ESX 2.x in VMware KB article 1290 and it would seem that this still applies to ESX 3.

Our task then, is to figure out what driver in the Console OS is actually using the interrupt on the same vector with the VMkernel. This is pretty easy with following command:

```
# cat /proc/vmware/pci | grep 0x71
Bus:Sl.F Vend:Dvid Subv:Subd Type Vendor ISA/irq/Vec P M Module Name
000:29.0 8086:24d2 1014:02dc USB Intel 11/ 16/0x71 A C
007:00.0 14e4:1659 1014:02c6 Ethernet Broadcom 11/ 16/0x71 A V tg3 vmnic1
008:00.0 14e4:1659 1014:02c6 Ethernet Broadcom 11/ 16/0x71 A V tg3 vmnic6
```

Of course you should replace 0x71 with the specific interrupt vector that is shown as shared on your system. Note that the "V" and "C" in the column labeled "M" (before the "Module" and "Name" columns) shows which vector is used by the VMkernel (V) or the Console OS (C). In my example the console OS is attached to the USB interface while the two onboard NICs are assigned to the VMkernel. So now we know that irq 16, which according to the output above corresponds with vector 0x71, attaches the USB hardware to the Console OS and the Broadcom nics to the VMkernel. If we want to know specifically which driver the Console OS is using to control the device we can simply run the following:

```
# cat /proc/interrupts
CPU0
0: 41270280 vmnix-edge timer
1: 4 vmnix-edge keyboard
2: 3075039 vmnix-edge VMnix interrupt
12: 15 vmnix-edge PS/2 Mouse
14: 5 vmnix-edge ide0
16: 76567 vmnix-level usb-uhci
18: 156 vmnix-level usb-uhci
19: 0 vmnix-level ehci-hcd
```

This shows that irq 16 is being using by the usb-uhci driver in the console OS.

Ideally at this point you would simply modify your systems BIOS settings to reallocate the USB hardware interrupt onto a different interrupt than the onboard NICs, or perhaps disable USB altogether if you don't need it. Unfortunately in the case of the IBM x336 there does not appear to be a way to change this. You can assign the onboard NIC and USB to a different IRQ, but that just moves the problem interrupt from one vector to another, since they still both share the same one. At first I thought of just disabling the onboard USB as I really didn't need it for anything, but I couldn't find such an option in the IBM BIOS.

So, thwarted by the system hardware I decided there were other options. First, I was pretty sure I didn't need USB support for normal operation so I thought I would simply disable the USB driver in the console OS. This is actually pretty simple, just run "modprobe -r usb-uhci" and it will unload the USB driver. As soon as you do this the output of the above commands change, look at the following:

```
#cat /proc/vmware/interrupts
Vector PCPU 0 PCPU 1 PCPU 2 PCPU 3
0x59: 0 0 0 0 <COS irq 13 (ISA edge)>
0x61: 1 0 0 0 <COS irq 14 (ISA edge)>
0x69: 0 0 0 0 <COS irq 15 (ISA edge)>
0x71: 68708836 1024 9813 7738 <COS irq 16 (PCI level)>, VMK vmnic6, VMK vmnic
0x79: 156 0 0 0 <COS irq 18 (PCI level)>
0x81: 0 0 0 0 <COS irq 19 (PCI level)>
0x91: 157878 658589 292915 741902 <COS irq 20 (PCI level)>, VMK ips
```

I edited this but the important part to note is the line with vector 0x71 now still show the VMkernel claiming the interrupts for vmnic1 and vmnic6, and that the Console OS has devices that it can claim; however, the presence of the angle brackets indicate that no driver is loaded by the Console OS for this device. You can also see that the interrupts have started being distributed across all CPUs. Also, the output of "cat /proc/interrupts" now shows no sign of interrupt 16 being claimed by any driver:

```
CPU0
0: 41365483 vmnix-edge timer
1: 4 vmnix-edge keyboard
2: 3083573 vmnix-edge VMnix interrupt
12: 15 vmnix-edge PS/2 Mouse
14: 5 vmnix-edge ide0
19: 0 vmnix-level ehci-hcd
NMI: 0
LOC: 0
ERR: 0
MIS: 0
```

After removing the driver we reran our benchmark and, sure enough, performance with the onboard Broadcom NICs were roughly equal to the performance of the Intel Pro/1000 adapter. If we ran "modprobe usb-uhci" performance would return to it's previously poor state, while unloading the driver with "modprobe -r usb-uhci" always returned the system back to good performance.

This proved that the usb-uhci driver was the culprit. We initially simply modified modules.conf to remove the usb-uhci driver completely, but that turned out to have a negative side effect that the IBM Remote Server Administrator card would no longer allow me to use the keyboard and mouse. Apparently the RSA emulates a USB keyboard and mouse and thus removing the driver removed the ability to use the remote keyboard. This really isn't that big of a problem since I usually just SSH into the system anyway, and if the system has crashed so hard that I can't ssh into it it's unlikely it will respond to the keyboard either. Plus, I can still use the RSA remote power features to power cycle the system and the keyboard works fine during the boot process.

Still, I decided to write a small script that simply unloads the usb-uhci driver 5 minutes after VMware ESX boots. That way if I'm attempting to troubleshoot something like a network connectivity issue, I can simply reboot the system, log in with the RSA and kill the script before the

five minutes are up. It's a cheap trick, but it actually works pretty well.

Another option I've thought about was to simply use the onboard NICs as standby adapters, so that they are only used if the Intel NIC malfunctions or loses its network connection. Not a bad option really, even at only 300Mb/s they will still likely survive the normal workday without any major issues.

I've even considered a more complex approach, something like running mon in the console OS, if it can't ping the default gateway, thus indicating a network problem, it could load the udb-uhci driver so that the RSA console would work, then, once the network is restored it could unload the driver.

Anyway, lots of options, but, in the end, it was an interesting experience and I learned a good bit about ESX server and how the VMkernel and Console OS interact with the hardware. Hopefully you'll find it interesting too.

---

Posted by Tom Sightler on Wednesday, November 22, 2006 at 14:50 in Linux

scikit

Trackback specific URI for this entry

PingBack

**Weblog:** [blog.joshcurrier.com](http://blog.joshcurrier.com)  
**Tracked:** Sep 01, 16:16

PingBack

**Weblog:** [blog.vmpros.nl](http://blog.vmpros.nl)  
**Tracked:** Sep 17, 13:44

PingBack

**Weblog:** [www.techhead.co.uk](http://www.techhead.co.uk)  
**Tracked:** Mar 20, 14:04

sting

Display comments as (Linear | Threaded)

katsumi liquer says,

Wednesday, January 17, 2007 at 14:32 (Reply)

holy shite is this a great article -- i have been searching for answers to my esx 3 and iscsi performance woes, and not even gold level VMware tech support is as good as you. GREAT JOB -- for real, you are a saint.. I don't know why you were so kind to share this knowledge, but it is greatly appreciated. I seemed to be having the exact same issue suing an HP Blade System c7000 chassis w/ BL 460c blades -- if you are ever in the boston area, please email me so I can buy you a drink.

Don Williams says,

Saturday, April 21, 2007 at 00:09 (Reply)

I agree great article.

There is a note on the VMware KB about problems with certain Broadcom 5700/5701 NICs.

<http://kb.vmware.com/selfservice/microsites/search.do?cmd=displayKC&externalId=2242>

Interesting about the USB driver. Recently saw a Vmware PSOD get fixed by upgrading the Firmware on a HP DL38x.  
The listed fix was the USB.  
Thanks again.

Tom says,

Tuesday, October 23, 2007 at 14:43 (Reply)

WOW, thanks a lot for sharing this info. Stumbling across your blog helped us to fix an issue that we've had since day 1 on one of our ESX boxes. By the way Vmware support repeatedly looked at our config and could not fix the problem.

THANK YOU.

Dejan says,

Wednesday, November 28, 2007 at 11:56 (Reply)

Great article.

I just could figure out why we had such a difference in performance on 3 identical (hardware wise) servers. I started looking into context switching problems and found your article by dumb luck.

I found that two of them had USB loaded and with the problems you described. And to make the situation worse the collided with RAID also (we use local discs)!

Again, thanks for a great article.

Tom Sightler says,

Wednesday, May 13. 2009 at 14:05 (Reply)

I'm sure you understand this, but because I've had so many strange questions about it I want to clarify that sharing interrupts doesn't mean there is a problem

If your NICs share interrupts with your RAID card, that's probably OK since both of those devices will be owned by the VMkernel. The problem occurs when an interrupt is shared by two or more physical devices and on some of those devices are controlled by the Console OS while others are controlled by the VMkernel.

The most common scenario seems to be the USB drivers because the VMkernel doesn't use the USB hardware but the Console OS does. If your USB controller (a device controlled by the Console OS) shares an interrupt with your NIC, or your storage controller (devices controlled by the VMkernel) then you're likely to see at least some performance penalty.

Collin C. MacMillan says,

Thursday, June 25. 2009 at 18:46 (Link) (Reply)

Likewise, if you're using ESXi then there is no VMkernel vs. VMSvc Console driver conflict because there is no VMSvc Console and the drivers belong 100% to the VMkernel.

A very good posting.

Tom Sightler says,

Sunday, June 28. 2009 at 20:22 (Link) (Reply)

Excellent point. We haven't even tried ESXi yet because we find the service console to be such a useful part of ESX we can't really imagine living without it. Actually, with VMware planning to do away with the service console we'll likely look to migrate away from VMware altogether. As a mid-size company (~1000 employees) we're just not really happy with their current technology direction and find their pricing to be out of line with the current market.

Timo Raatikainen says,

Tuesday, January 22. 2008 at 17:22 (Reply)

This is just great!

This article helped me to solve my problem with network throughput completely! We were getting 10MB/sec until I unloaded the USB drivers. Thanks!

Mike La Spina says,

Sunday, April 27. 2008 at 11:17 (Link) (Reply)

Excellent blog, thanks for sharing what you have learned.

Frank says,

Monday, December 22. 2008 at 03:42 (Reply)

Hi - could you please post your script you use to unload the drivers?? and instructions how to kill it? We have the same problem on HP hardware

John says,

Thursday, April 16. 2009 at 09:10 (Reply)

Would love to see the script too (or could you email it to me).

Tom Sightler says,

Monday, May 4. 2009 at 18:20 (Reply)

The reason I didn't post the script is because it unloads only the driver that we needed to work around this problem for our particular hardware. It's very possible that the same driver is causing your problems, but it's just as likely (and probably more likely) to be a different driver. Not only that, but the script is a whopping 2 whole lines of code (the rest are comments).

I pretty much assumed that anyone charged with administering VMware ESX servers would be able to write a two line script. Heck, is two lines really even a script? You could just as easily throw these two commands in rc.local, but we typically create scripts with names and comments so we get hinted as to what they do, our actual script has a more detailed comment explaining why we even need such a stupid script in the first place.

```
#!/bin/sh
#
# stopusb
#
# This is a stupid script to simply sleep for 300 seconds
# after boot and then unload the usb-uhci driver from the
# service console.

sleep 300
/sbin/modprobe -r usb-uhci
```

Patrice B says,

Tuesday, May 12. 2009 at 20:14 (Link) (Reply)

Can this `/sbin/modprobe -r usb-uhci` can be execute from a remote site without any problem to the server ?

I mean, is that command can affect the working users if I doing that from remote site ?

Tom Sightler says,

Wednesday, May 13. 2009 at 13:37 (Reply)

The command should have no affect on users of the VMs as it doesn't really have anything to do with the virtual machines, it is simply unloading the USB drivers from the console OS. If your host machine uses some USB device, like a keyboard or mouse, those may stop working, so you wouldn't be able to access the console of the ESX server with a keyboard, but you would still be able to access the system via ssh, and remotely access any of the VMs with the remote console. If you need to use a local USB keyboard temporarily you can always ssh to the console and reload the USB driver with:

```
/sbin/modprobe -r usb-uhci
```

Patrice B says,

Wednesday, May 13. 2009 at 17:58 (Link) (Reply)

I did that but did fix anything for me. Still very slow  
It's an IBMx3500

I know if I go into the Infrastructure utility, I can see that a USB as been installed for the VM machine but says it's not supported, is it possible that cause the slow transfer event if i did the `modprobe -r usb-uhci` ?

Or do you have any other suggestions?

Tom Sightler says,

Wednesday, May 13. 2009 at 22:27 (Reply)

Did you actually read the article and perform the troubleshooting steps or are you just trying my resolution assuming that you have the same problem I did with the exact same driver? While I published the solution to my problem on my specific hardware, that's not the point of the article.

This article is written to teach you how to determine if you are having an interrupt sharing conflict between the Console OS and the VMkernel and, if so, determine which driver is causing the conflict. If you are having an interrupt sharing issue you must then determine how to resolve it, either by unloading a driver or perhaps reconfiguring your hardware, sometimes simply moving cards around will work. If you're not having an interrupt conflict then nothing in this article is going to help.

So, in summary, did you first determine that you were having an interrupt problem before you rushed to trying to unload the driver? If not, you should do that first.

Maik says,

Wednesday, May 13. 2009 at 10:40 (Reply)

You saved my day. Thank you very much. This isn't limited to Broadcom - I've been battling my Intel Dual Port cards here for days, wondering about their abysmal performance. Lo and behold - they shared the same interrupt as the console usb driver and were bound to CPU0. Now this iSCSI setup flies... Thanks again!

Tom Sightler says,

Wednesday, May 13. 2009 at 13:54 (Reply)

Right, the problem certainly isn't limited to Broadcom, it can actually affect any device where a physical interrupt is shared between a driver in the console OS and a driver in the VMkernel. I've even seen at least one instance where it impacted disk performance because the USB driver was sharing an interrupt with the fiber channel controller. The performance impact in this case was less noticeable (around 15-20%) but easily measurable.

Glenn H. Eriksen says,

Friday, May 15. 2009 at 11:38 (Link) (Reply)

Wanted to share this new KB article from VMware with everyone:

<http://kb.vmware.com/selfservice/viewContent.do?externalId=1003710&slcid=2#determine>

This explain the exact same problem, but update for ESX 3.5, and with some new info.

I have the same problem with a couple of Intel Server with Quad NICs and HBA cards, but my conflict is with the COS Keyboard! So I can't just deactivate it... I'll try disabling stuff in the BIOS, along with moving NIC and HBA cards to different slots to see. But I'm sure it will be resolved some how. Thanks anyway for pointing my strange problem in the right direction!

Ron Neilly says,

Tuesday, June 9. 2009 at 21:14 (Reply)

Thanks for documenting this and doing so clearly. It allowed me to identify a problem with the ehci (enhanced usb) device being loaded by the service console on the same interrupt as many of our nic's and hba's.

Instead of scripting the unloading of the device driver I simply commented out the load line in `/etc/modules.conf`.

Cheers.

Tom Sightler says,

Tuesday, June 9. 2009 at 22:06 (Reply)

Glad it helped. Removing/commenting the line from modules.conf is a very valid solution. The only reason I used a delayed scripting solution for our servers was that removing the uhci driver caused the remote access feature to not have keyboard control. I use the 5 minute delay as a sort of "escape hatch". Basically, if something happens bad enough that I need to use the remote access console (i.e. loss of SSH network access to the service console) I can reboot the server, and then I have five minutes to login with the remote access console and kill the "unload script". It's probably overkill, but it saved me once on a server that was an ocean away, so I still use this method.

seb says,

Monday, June 29. 2009 at 06:30 (Reply)

Hi,

and how to apply this on esxi? AFAIK there is no modprobe... thanks!

seb

Tom Sightler says,

Wednesday, July 1. 2009 at 20:55 (Reply)

ESXi should cannot have the problem described in this particular article since there is no service console. The VMkernel owns all interrupts.

That doesn't mean that ESXi couldn't have a network performance problem, but it wouldn't be caused by an interrupt problem with the service console.

userone says,

Friday, July 3. 2009 at 14:57 (Reply)

It looks like that ESX 4 (vSphere) might not have this issue  
See following from a HP DL380 G5/ESX 4 (Hope it is true):

```
Vector PCPU 0 PCPU 1 PCPU 2 PCPU 3 PCPU 4 PCPU 5 PCPU 6 PCPU 7
0x21: 0 0 0 0 0 0 0 0 VMK ACPI Interrupt
0x22: 2246551 1584882 473506 1230378 2258795 1698829 2965905 3903666 VMK vmnic6
0x29: 1 0 0 0 0 0 0 0 CCS irq 1 (ISA edge)
0x2a: 92310 197625 37995 65280 421543 91545 117300 51510 VMK vmnic7
0x31: 3 0 0 0 0 0 0 0
0x32: 181077 216495 173145 136425 78285 65792 89505 134385 VMK vmnic8
0x39: 5 0 0 0 0 0 0 0
0x3a: 49470 155550 41849 116535 372300 134130 49980 155295 VMK vmnic9
0x41: 1 0 0 0 0 0 0 0
0x49: 1 0 0 0 0 0 0 0 CCS irq 8 (ISA edge)
0x51: 0 0 0 0 0 0 0 0
0x52: 2903018 1793928 1995905 1969202 929731 1062336 1156449 1099319 VMK cciss0
0x59: 94 0 0 0 0 0 0 0 CCS irq 12 (ISA edge)
0x61: 0 0 0 0 0 0 0 0
0x69: 75888 72676 58396 74970 65537 53041 56100 71910 , VMK libata
0x71: 0 1 0 0 0 0 0 0 , VMK libata
0x79: 0 0 0 0 0 0 0 0 , VMK uhci_hcd:usb2, VMK ehci_hcd:usb1
0x81: 18987345 11599593 21037269 23424302 22144712 22630377 21186424 21527883 , VMK lpfc820, VMK uhci_hcd:usb3
0x89: 89505 125775 12623 76118 39015 63368 64388 28560 , VMK lpfc820, VMK uhci_hcd:usb4
0x91: 0 0 0 0 0 0 0 0 , VMK uhci_hcd:usb5
0x99: 0 0 0 0 0 0 0 0
0xa1: 0 0 0 0 0 0 0 0
0xa9: 76 0 0 0 0 0 0 0 , VMK uhci_hcd:usb6
0xb1: 0 0 424671144 0 0 0 0 0 VMK vmnic0
0xb9: 0 0 15887664 0 0 0 0 0 VMK vmnic1
0xc1: 17150540 7276352 17066888 14817717 17940338 18078803 92458769 91250704 VMK vmnic2
0xc9: 2963036 2342438 75735 632658 187425 198394 1087830 1232417 VMK vmnic3
0xd1: 4119015 3178263 385306 1056721 188955 260355 539332 622710 VMK vmnic4
0xd9: 96900 378165 116790 117839 73950 126225 69360 95880 VMK vmnic5
0xdf: 1140618460 1220555952 1257201109 1260270679 1283699480 1285351766 1284983403 1286730325 VMK timer
0xe1: 393180 2941670 3891808 4389559 3531838 3701197 3222568 3274408 VMK monitor
0xe9: 570608175 737958186 354096437 378296109 372263598 372007246 382907516 382532491 VMK resched
0xf1: 46980 262720 283068 267401 299996 285801 280138 273801 VMK tlb
0xf9: 65053074 0 0 0 0 0 0 0 VMK noop
0xfc: 0 0 0 0 0 0 0 0 VMK thermal
0xfd: 0 0 0 0 0 0 0 0 VMK lint1
0xfe: 0 0 0 0 0 0 0 0 VMK error
0xff: 0 0 0 0 0 0 0 0 VMK spurious
```

Tom Sightler says,

Friday, July 3. 2009 at 18:05 (Reply)

Not sure, haven't played with vSphere4 yet, but in the example above it appear that the USB drivers are owned by the VMkernel rather than the Console OS which would eliminate the most common cause of the problem.

That being said, something sure is suspicious about vmnic0 and vmnic1. There's no interrupt sharing going on there at all, which would imply the problem might still exist. The output of /proc/vmware/pci and /proc/interrupts would be interesting.

seb says,

Saturday, July 4. 2009 at 17:12 (Reply)

Hi Tom,



this is strange - I encounter exactly the same problems (very low iscsi speed).

May I link to my thread with screenshots?

I still have no idea what to do.

<http://communities.vmware.com/thread/218231?tstart=0>

Tom Sightler says,

Saturday, July 4, 2009 at 18:19 (Reply)

Following the post, however, I don't think you have the problem in this paper. According to the thread you're running ESXi, it can't really have the problem described in this thread. It's possible it may have other problems.

Still, in perusing your thread you seem to be trying to connect your ESXi system to some non-certified desktop iSCSI box because you say "it's iSCSI and ESXi supports iSCSI". Well, ESXi does support iSCSI, but it expects iSCSI devices to support enterprise features like SCSI reservations and battery backed write back cache (so that synchronous writes don't cause full cache flushes which hammer performance, especially on writes). These desktop style boxes do "support iSCSI" but they're not enterprise iSCSI solutions and are not on the ESXi HCL thus they are not certified or supported. They might work, but probably not well. Even many enterprise iSCSI arrays have suffered from performance problems when used with iSCSI because of high overhead for synchronous write or poor performance with iSCSI reservations.

You can work around many of these issues by using the iSCSI initiator from within your guest VMs.

userone says,

Tuesday, July 7, 2009 at 13:06 (Reply)

y, you are right---the vmnic0 and vmnic1 (d380 on board NICs) might still have the problem but I could not find what is the cause.

See the /proc/interrupts:

```
# cat /proc/interrupts
CPU0
0: 248973570 timer
1: 10 i8042
2: 519808144 VMxix interrupt
8: 1 rtc
12: 104 i8042
NMI: 0
LOC: 0
ERR: 0
MS: 0
```

On ESX 4 (vSphere), they moved the /proc/vmware/poi (at least I could find it in /proc/vmware/), they also remove the modules.conf from /etc.

could not find any doc for this. Looks like they did change many ESX4's code.

Andrew says,

Tuesday, August 4, 2009 at 07:49 (Reply)

That's correct. The VMKernel owns and runs all device drivers in vSphere. There is no longer any issue with drivers being pinned to CPU0.

Tom Sightler says,

Tuesday, August 4, 2009 at 16:01 (Link) (Reply)

I suspected as much, but it's nice to know that's the truth. I hope to start tinkering with vSphere in the next few weeks.

6 dA

Name

Email

Homepage

In reply to

[ Top level ]

Comment

Enclosing asterisks marks text as bold (\*word\*), underscore are made via \_word\_.  
Standard emoticons like :- ) and :- ) are converted to images.

To prevent automated Bots from comment spamming, please enter the string you see in the image below in the

appropriate input box. Your comment will only be submitted if the strings match. Please ensure that your browser supports and accepts cookies, or your comment cannot be verified correctly.



Enter the string from the spam-prevention image above:

☐ Remember Information?

Submitted comments will be subject to moderation before being displayed.

© 2009 by Tom Sightler | Design by Andreas Viklund | Serendipity template by Carl and Bex