

Utilizing Intel® Virtualization Technology on an Intel® 5000 Chipset Series-based Platform

White Paper

April 2007



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting [Intel's Web Site](#).

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor_number for details.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino logo, Core Inside, FlashFile, i960, InstantIP, Intel, Intel logo, Intel386, Intel486, Intel740, IntelDX2, IntelDX4, IntelSX2, Intel Core, Intel Inside, Intel Inside logo, Intel Leap ahead., Intel Leap ahead. logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, Itanium, Itanium Inside, MCS, MMX, Oplus, OverDrive, PDCharm, Pentium, Pentium Inside, skool, Sound Mark, The Journey Inside, VTune, Xeon, and Xeon Inside are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2007, Intel Corporation. All rights reserved.

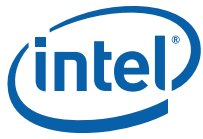


Contents

Introduction	5
Overview	5
Demo Procedure	10
Conclusion	17
References	17
Appendix A - Demonstrating Processor Affinity	17

Figures

1	Hardware Consolidation for many Virtual Machines	6
2	Manager Server and Target System	7
3	Hardware configuration of the target machine discovered by Virtual Iron* Virtualization Manager	9
4	Pin a virtual machine to a specific physical processor core	13
5	Execute command to start virtual machine to run on the specified processor core	13
6	Virtual Machine Device Manger view	15
7	VM1 Disk Management view	16



Revision History

Date	Revision	Description
April 2007	001	Initial release.



Introduction

System architects in the Server and Workstation Multiprocessing environments have been actively using Virtualization Technology to realize cost savings through hardware consolidation, fast fail-over mechanisms (without extra hardware), and other beneficial use models. Now this technology has increased interest in the embedded and communication markets as well.

In the past, embedded and communication markets have not been able to make proper use of virtualization, given the overhead associated with the extra layers of software required for such setups. We now have hardware support for virtualization in processors and chipsets. Intel has implemented a set of accelerations called Intel® Virtualization Technology, or Intel® VT, that addresses some of the challenges and drawbacks of software-only solutions, opening the benefits of virtualization to a much wider audience.

Intel® Virtualization Technology (Intel® VT) for IA-32, in particular Intel VT-x (available now), allows a Virtual Machine Monitor (VMM) to run unmodified guest Operating Systems (OSs) with increased performance and reliability (when compared to the software only solutions). This is accomplished by reducing the frequency in which a VMM needs to interlude during normal run-time operation.

Future releases, such as Intel® Virtualization Technology (Intel® VT) for Directed I/O (Intel® VT-d) , will further reduce the barriers to the adoption of virtualized hardware solutions. Go to <http://www.intel.com> or talk to your Intel representative for more information on Intel Virtualization Technology.

This paper focuses on one common use case for virtualization in the embedded space, demonstrating a proof-of-concept for a solution where the processing resources of many computer terminals have been consolidated to a single hardware platform via a VMM providing many virtualized machines.

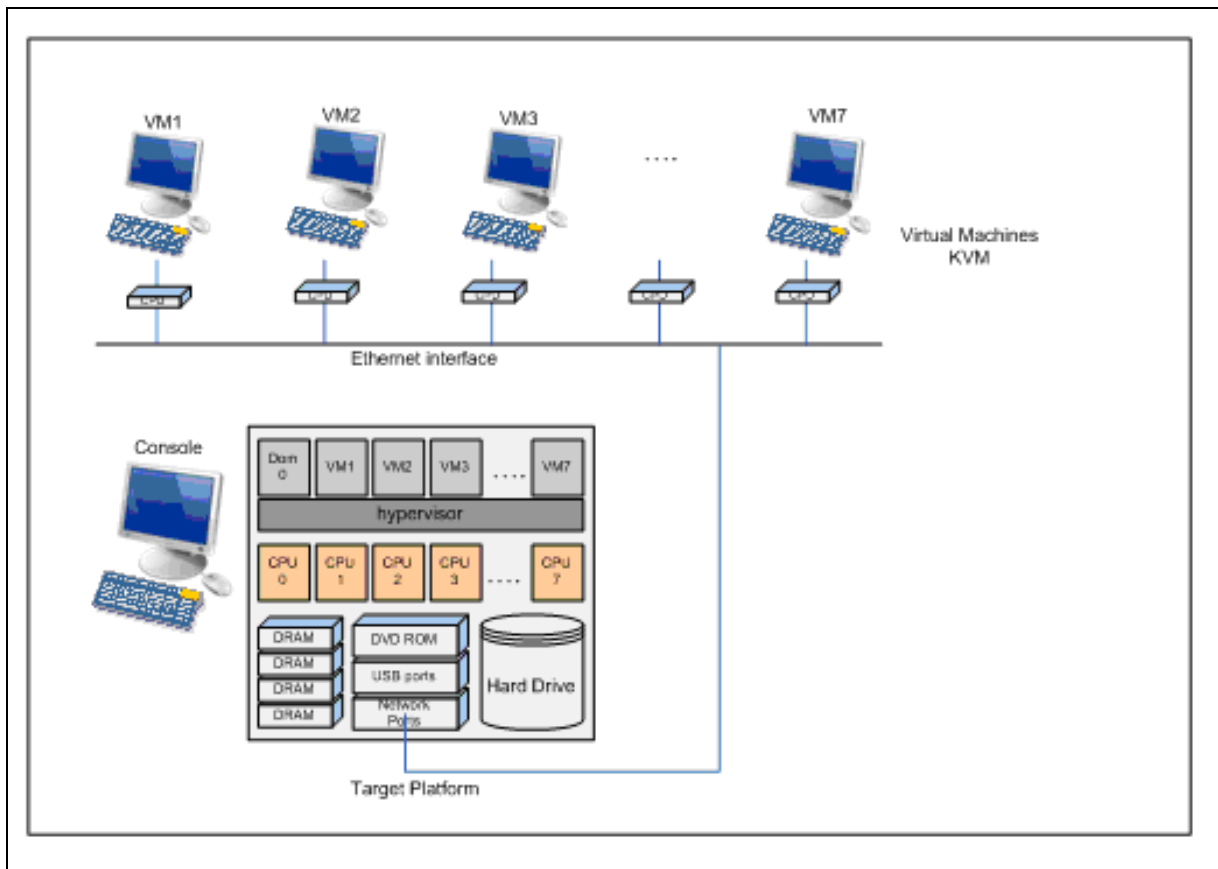
Overview

The goal of this paper is to provide step-by-step instructions to create a proof-of-concept system with the functionality described in [Figure 1](#).

In the embedded environment, the applications typically allocate hardware resources up front, hence it is important for the VMM to be able to allocate specific hardware resource, such as process units and memory, to virtual machines. This paper demonstrates how to tie a virtual machine to a specific processing unit using the VMM available in the Server market segment. Furthermore, this demonstration establishes seven virtual machines that share a common processing base, but can be operated independently with their own processing unit and I/O resources, such as a monitor, keyboard, and mouse. This type of setup would effectively consolidate the most expensive and potentially most under utilized piece of each system, the processor/chipset combination. Over the life of the product, more money is saved in operating costs such as less energy consumption (fewer platforms to power) and service and repair (single place to access multiple virtual machines).



Figure 1. Hardware Consolidation for many Virtual Machines



Since Intel VT is so new, not all VMM vendors support it. Many VMMs targeted towards the embedded and communication space do not have fully-supported versions of their products available off-the-shelf. This demonstration uses software that is more typically found in other market segments (like Server) since they are more mature (Virtual Iron* 3.1 is used in this demo).

The following sections describe the hardware and software required to set up this virtualization proof-of-concept. Following that, you will find step-by-step instructions to put together and test it.

Required Hardware

This demonstration includes several elements, each with their own hardware requirements:

- a target platform
- a management server platform
- several "remote" machines
- a few network elements to connect them all together

In general, you will need all the equipment listed; if there are optional pieces, they will be marked as such.



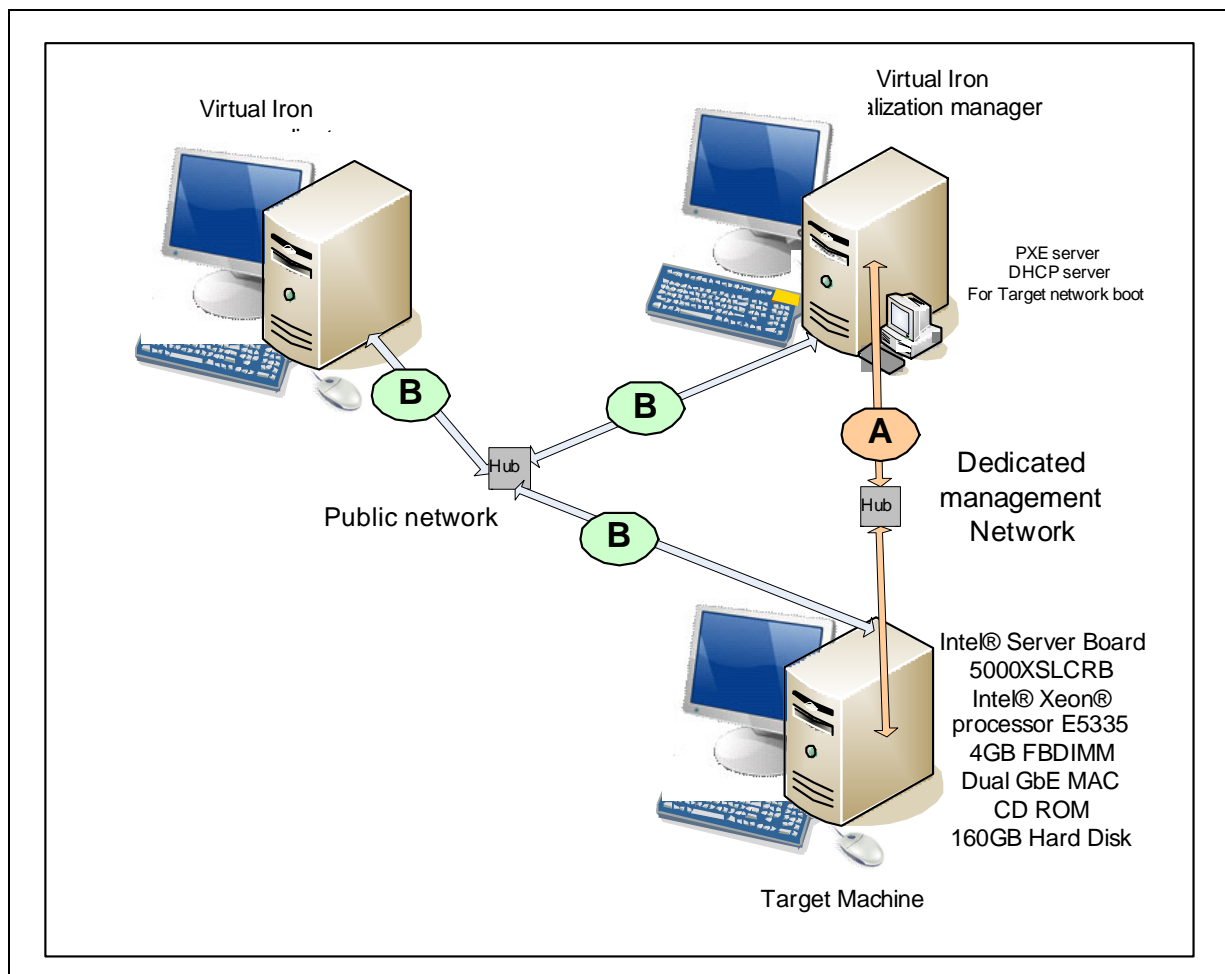
Management Server Hardware

The Virtual Iron infrastructure requires a single manager server linked to one or more managed (target) nodes over a dedicated network. The management server is an x86 system running either Linux* or Windows* XP while the target nodes do not run a standard OS.

The target nodes are booted through a network boot via the dedicated network interface where the PXE and DHCP services are provided in the management server. The management server can be accessed by remote clients through a public network. Virtual Iron allows many target nodes to be managed by one virtualization manager node. However in this demonstration, only one target system is deployed.

Figure 2 shows the Virtualization Manager Machine connected to the Target Machine (managed node) via a dedicated management network interface as denoted as link A. The other machines are connected via public network link B in Virtual Iron virtualization environment. The various network elements (hubs, cables, etc.) will be described in another section below.

Figure 2. Manager Server and Target System



In the development of this demo, a Dual-Core Intel® Xeon® processor LV and Intel® 3100 chipset were used as the manager server system. Any x86 systems with two network ports, CDROM, and 2 GB of memory can be used as the manager server system.



The management hardware platform consists:

- Any x86 system
- 2 Ethernet ports
- 1 CDROM drive
- 2 GB RAM
- 30GB Hard Drive

Target Platform Hardware

The target hardware platform used to implement this demo is the Intel® Server Board S5000XSL-based CRB (Customer Reference Board) with two Quad-Core Intel® Xeon® Processors E5335, for a total of eight processing units available in one system. These are connected to the Intel® 5000X Chipset Memory Controller Hub via a shared Front Side Bus running at 1333 MTS.

The target system includes 2 built-in Gigabit Ethernet MAC with Intel® 82563EB Gigabit Ethernet Controller. With Intel® I/O Acceleration Technology implemented in the Intel Xeon processor E5335 the target system moves network data more efficiently across diverse operating systems and virtualized environments.

4 GB of Fully Buffered DIMMs (FBDIMMs) is installed on the platform to be used for the VMM and VMs. The FBDIMMs provide higher memory density to the system while still using a conventional DIMM package. A 160GB SATA hard drive is installed for local storage of all virtual machine activity. A monitor and USB keyboard and mouse are connected for the main target.

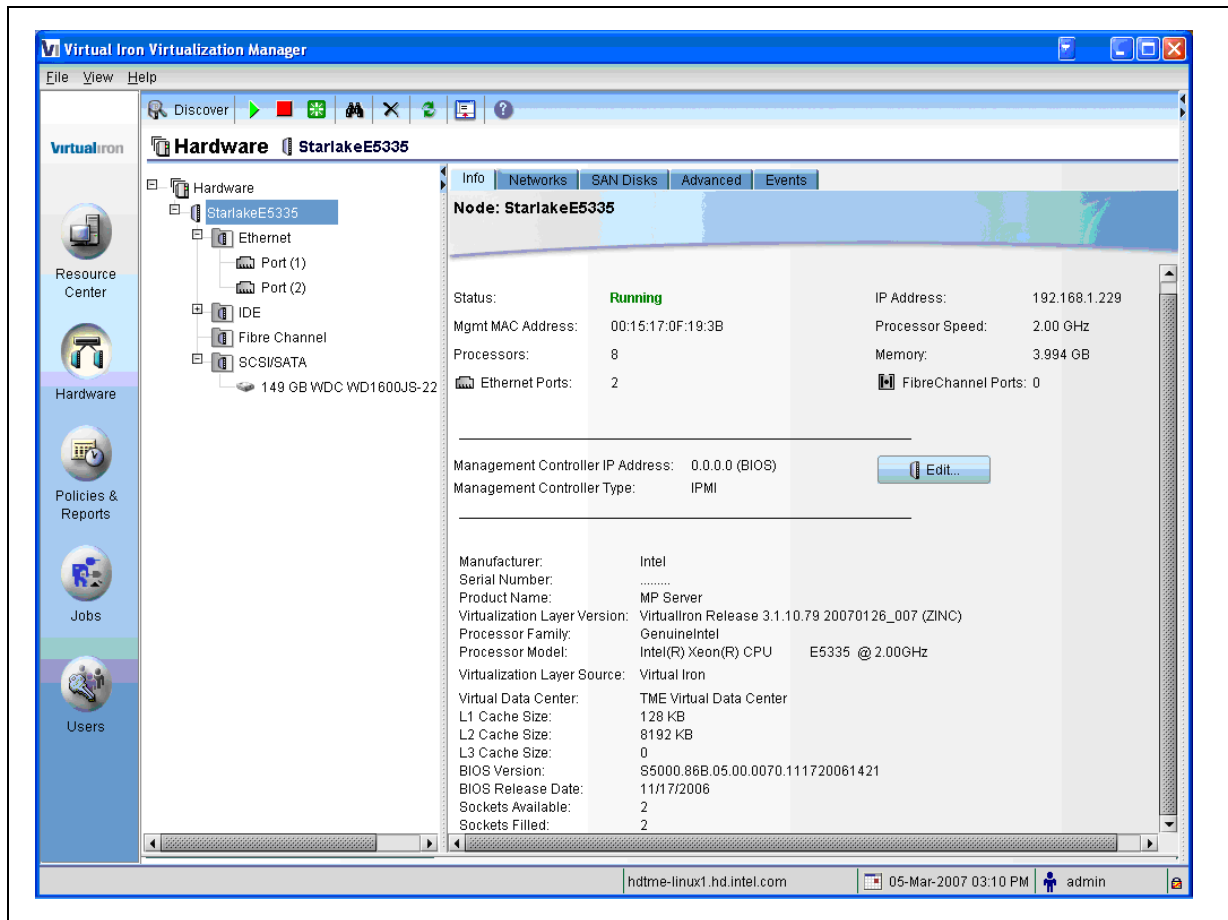
Be sure the Intel Server Board S5000XSL BIOS is at least version S5000.86B.05.00.0070 as shown in [Figure 3](#) so that the quad core processors are recognized and the Intel VT-x based hardware accelerations are enabled.

A summary of the target hardware platform:

- Intel Server Board S5000XSL
- Two Quad-Core Intel® Xeon® Processors E5335
- 4 GB of FBDIMM
- 160 MB SATA Hard Drive
- One CDROM Drive
- One monitor
- One USB keyboard
- One USB mouse

The figure below shows the readout of the actual target machine discovered by the Virtual Iron Manager.

Figure 3. Hardware configuration of the target machine discovered by Virtual Iron* Virtualization Manager



Manager Client Hardware

Any system with a web browser running Java* 1.5.0_10 can be used to access the manager server remotely as long as the system is in the network. Access is protected by the user name and password authentication.

Network Elements

Two Linksys* 16-port Workgroup hubs are used to connect the private management network and public network as shown in Figure 2. At a minimum, the hubs should have 10 ports and be able to support 100 Mbps per port. Switches with similar configurations would be even more desirable.

- Two Linksys 16 port Workgroup hubs

In addition, a minimum of 6 Cat5 Ethernet cables will be required.

- Four Cat5 Ethernet cables for the target and management network setup
- Two to seven Cat5 Ethernet cables (one per remote machine)



Remote Machines

The processor, memory and data storage for each remote machine will be made available as part of a "virtual machine", running on the target platform described earlier. Only user-interaction I/O (display, keyboard, mouse) will be required "extras" for each remote machine.

- Two to seven displays (one per remote machine)
- Two to seven keyboards (one per remote machine)
- Two to seven mice (one per remote machine)
- Two to seven KVM network appliances (one per remote machine)

In the development of this demo, Digi ConnectPort* Display over IP (CPD) was used as the KVM network appliance. This unit contains a SVGA/VGA port, two USB and two serial ports that allow the KVM connection directly to the Ethernet without being locally attached to a PC.

The KVM network devices will be connected via Cat5 Ethernet cable to the public network of [Figure 2](#) via the instructions in the Procedures section.

Note: When setting up your proof-of-concept, start by setting up two remote machines successfully, then add more as desired.

Required Software

This demo uses some common off-the-shelf software packages, including Red Hat* Enterprise Linux 4 (update 4) on the Management Server Platform and Windows XP* for some of the virtual machines. Virtual Iron virtualization software will be used to partition the hardware resources of the target platform into virtual machines for each of the Remote Machines. Beyond the software to set up the system, some applications such as Windows Media Player* and the games that come with Windows XP installation, can be used to test whether the remote machines are working properly.

Here is a full list of the required software:

- Red Hat Enterprise Linux 4 - update 4
- Windows XP Professional with Service Pack 2
- Java Runtime Version 1.5.0_10
- Virtual Iron 3.1
- Digi ConnectPort* software
- Windows Media Player 9
- CPUBENCH V4.0.0.6

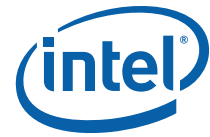
While other VMMs and OS combos were not tested in the creation of this paper, they could potentially be utilized as long as the VMM supports Intel VT-x, multi-core systems and the desired operating system(s). However, the exact detail of these alternate software setups is beyond the scope of this document.

Demo Procedure

Procure the hardware and software listed in the above sections before moving on to the procedures below.

Hardware Setup

The hardware setup mainly consists of the management server and target platform, and the network configuration between the two systems.



Management Server Platform

1. If you are using a brand new system, set it up using the instructions that came with it. Be sure to install a two-port network card if the system does not have a built-in NIC.
2. Connect the management server to the private network hub via Ethernet cable using one of the network ports on the systems. Connect the other network port to the public network hub as shown in [Figure 2](#).
3. Install Red Hat Enterprise Linux 4-U4 (Windows XP is also supported), configure the network interfaces with static IP addresses - one private, one public.
4. Once the RHEL4 is installed, follow the Virtual Iron user's guide to install Virtual Iron 3.1 and choose "separate public and management networks" network setup. You will need to have the network configured successfully in step 3 to complete the Virtual Iron* installation.
5. After the Virtual Iron Install, reboot the Manager Server system to make sure the management service is started. From then on, you can access the manager server either from the Management Server system or any other Manger Client system by entering the public IP address assigned to the manager Server System with a web browser and password.

Target Platform

1. Be sure to install the processors, memory, hard drive, CD-ROM drive, display, keyboard and mouse as specified in the Users Guide.
2. Connect the target system to the private network hub via Ethernet cable using one of the built-in network ports on the systems. Connect the other network port to the public network hub as shown in [Figure 2](#).
3. Connect the power supply to the board. At this point, you should be able to boot your platform, but, you will need to use a jumper to activate the FP_PWR_BTN_N (pin 11) of the front panel connector (J1E4) and power on the board.
4. When the system comes up, press F2 to enter BIOS setting and verify the BIOS version. Make sure Intel VT is enabled and network boot precedes other methods in the boot sequence.
5. The target platform boots the VirtualizationServices.iso image received from the Manager Server during network boot and can be managed with the Virtualization Manager client. You will see the screen is flashed with a lot of boot messages, and will end with a message like "Node 192.168.1.229 is awaiting discovering." This indicates that the target node is ready to be managed.

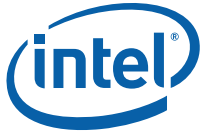
Network Setup

Reference [Figure 2](#) for a diagram of what is meant by "private management network" and "public network". Basically the private management network is the one between the Virtual Iron Manger Server platform and Target platform, while the public network is the one that connects to the world.

1. Connect the Target platform, Manager Server Platform to the private network hub.
2. Connect the KVM network appliances, the Target Platform and Manager Server Platform to the public network.
3. For each "remote terminal", connect the monitor, keyboard and mouse to the VGA port and the two USB ports of the ConnectPort Display unit respectively. To start, let's connect enough I/O for 2 remote terminals. You can add more later.

VMM Software Configuration

At this point, the management server can be accessed remotely by any system with a web browser that has Java 1.5.0 _10 installed. Through the Java based user interface, user can create a virtual machine, configure memory, storage and network interface and install new OS easily.



Starting a New Virtual Machine

1. Start the Virtual Iron Virtualization Manager on the management system using the internet browser by typing in the IP address of the Manager system (for example: 10.127.142.41 if you've configured your static IP address as mentioned in the previous sections).
2. Click on the Hardware in the left to check the target hardware resources discovered and make sure both network ports are up.
3. Click on the Resource Center and configure these items as described in the Virtual Iron User's Guide:
 - Create a virtual machine
 - Assign virtual NIC and virtual disc
 - Configure memoryFor example:
VNIC(MAC Address)=VNIC-00:0f:4b:03:ce:4C
Memory = 512MB
Assigned Disks = 6000F4B00000000001105123a0ec
4. Start the virtual machine by right clicking on the green "start" button.
5. Finally, install the Windows XP Operating System by placing the CD in the target system.
6. Repeat step 3 to 5 for the second VM.
7. If you are adding more than two remote displays to the demo, repeat as necessary.

Pinning a VM to a Core

For this demo each virtual machine can be pinned to run on a particular core in our hardware. Processor affinity is supported in Virtual Iron 3.1 but is not exposed to the GUI interface. So we will have to make change by hand in the Node.Name.xml file for each virtual machine.

When a VM starts, the manager creates this file in the directory where Virtual Iron Manager is installed; if you followed the installation procedure above, this directory should be created in /opt/VirtuallIron/VirtualizaitonManager/bootfiles/configuration/.

1. Shut down the virtual machine that you just started by right click on the red shutdown button of the virtual machine so that we can edit this Node.StarlakeE5335.xml file.
2. For the first VM, copy the Node.StarlakeE5335.xml file and rename it Node.StarlakeE5335-vs1.xml
3. Modify the xml based configuration for the first VM by adding the following line to the Node.StarlakeE5335-vs1.xml file.
<VCPU ID="1" Proc="1"/>

Between the "<Processor>" and "<Current>1<Current>" tags, the Proc="1" tag indicates that physical core 1 should be used for this particular virtual machine. In our 8 core system, this tag can have a value from 1 to 7 as core 0 is reserved for the VMM (domain 0).

At this point, the file should look something like [Figure 4](#) below:

Figure 4. Pin a virtual machine to a specific physical processor core

```
<Domain ID="1" SeqNo="1">
  <Name>TME-VS1</Name>
  <Processor>
    <VCPU ID="1" Proc="2"/> //Proc=2 means Domain 1 will run on physical processor 1
    <Current>1</Current>
    <Max>1</Max>
  </Processor>
```

4. After the xml is modified, go to /opt/VirtualIron/VirtualizationManager/support_tools and execute the following commands to restart the virtual machine.

```
./testagent --exec="/etc/xen/scripts/network-bridge start bridge=networkname=eth0"
TargetNode_IP
```

```
./testagent --configure=tftp://Manager_node_IP/conFfiguration/Node.xml TargetNode_IP
```

Figure 5. Execute command to start virtual machine to run on the specified processor core

```
[root@hdtme-linux1 support_tools]# ./testagent --exec="/etc/xen/scripts/networkbridge start bridge=tmeLabNet netdev=eth0"
192.168.1.229
Status: 200 Started - PID 10621

[root@hdtme-linux1 support_tools]# ./testagent --configure=tftp://192.168.1.101/configuration/Node.StarlakeE5335-vs1.xml
192.168.1.229
Status: 200 Success
```

As you can see, the manager server is executing the command to the target node using the private IP address (192.168.1.229 is the target node IP). After the "status: 200" success message, you will see the VS1 turn green in the GUI, indicating that the first virtual machine with its own OS is running. See ["Appendix A - Demonstrating Processor Affinity"](#) for more information about how the pinning of a VM to a particular Core works.

5. Repeat steps 2-4 for the second VM with the following modifications:
 - Use Node.StarlakeE5335-vs2.xml as the name of the file in step 2
 - Use Proc="2" as the tag to add in step 3
 - Use Node.StarlakeE5335-vs2.xml as the file name in the second command of step 4.
6. If you are adding more than two Remote machines to the demo, repeat this section as necessary.



Assigning I/O to Remote Machines

1. Install Digi ConnectPort software on the first virtual machine, using the instructions that come with it.
 - Assign the ConnectPort Display with a static IP address during the installation.
 - Make sure UltraVNC service is running on the virtual machine and password is set.
2. After the install, bring up a web browser on the virtual machine and enter the IP address of the CPD. The "ConnectPort Display M22 Configuration and Management" page will allow you to edit the configuration and reboot system. If everything is set up correctly in the configuration, you should now be able to see the remote display on the monitor connected to the CPD.
3. Use the mouse and keyboard connected to this KVM to make sure they work properly.

Testing the Demo Setup

Now that the virtual machines are running, verify that each VM discovers the system resources assigned to it using the Virtual Iron configuration. The system configuration of each VM should match the settings we specified in the Virtual Iron interface in previous sections.

Further functional testing can then start by installing applications on each of the virtual machines. For example, it will be important to test independence of the VMs from a processing or security point of view. Each VM should be able to run applications or crash and not affect the other VMs. In the sample scenario described below, we run processor intensive benchmark tests on each VM individually and then compare these results to running these benchmarks on each VM at the same time. We should see that the time it takes to run the benchmarks on each VMs does not increase, even as the other VMs are also exhausting their allocated CPU resources. The purpose of running CPUBENCH is to put the processor cores on the treadmill resulting in high CPU usage for each VM, not to collect the benchmarking data.

Verify Virtual Machine resources

1. In VM1, use the mouse connected to the CPD to right click on "My Computer". From the pop-up window, click on "Manage" to bring up a display of the resources that have been assigned to VM1.
2. Click on "Device Manager" from the left-hand tree under "System Tools" and take a look at the system resources in the right window pane.
3. Specifically, click on "Processors" and note what processor type and speed are listed. Verify that this matches the Virtual Iron configuration for this virtual machine. This is highlighted in [Figure 6](#).
4. Next, click on the "Disk Management" from the left-hand tree under the "Storage" to see if disk space matches the configuration specified during the creation of the VM. See [Figure 7](#) for example.
5. Finally, verify the memory assigned to the VM by right-clicking on "My Computer" and selecting "Properties" from the pop-up window. You should see the "Intel® Xeon® CPU E5335 at 2.0 GHz, 512 MB of RAM, Physical Address Extension" listed under "General" panel of the System Properties.

Figure 6. Virtual Machine Device Manger view

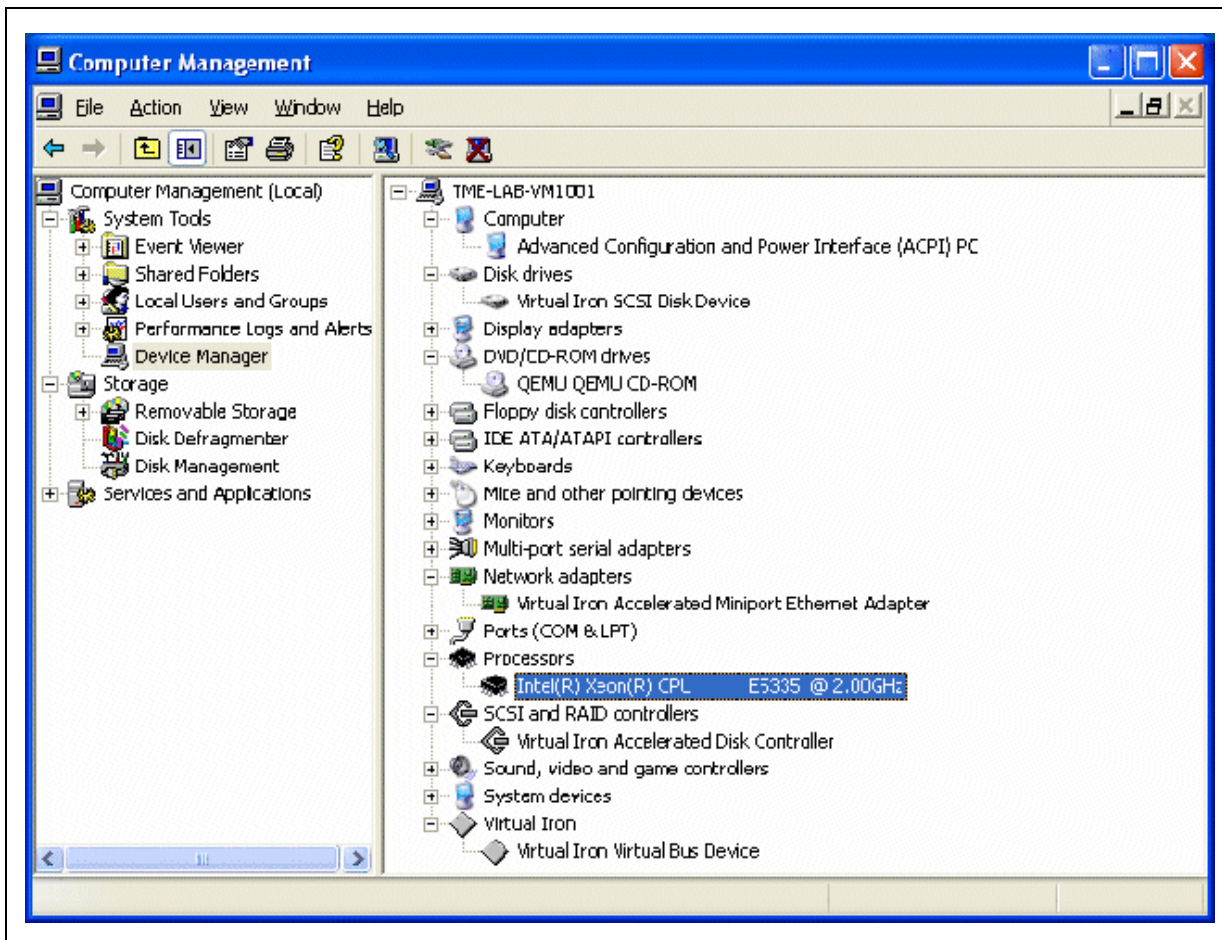
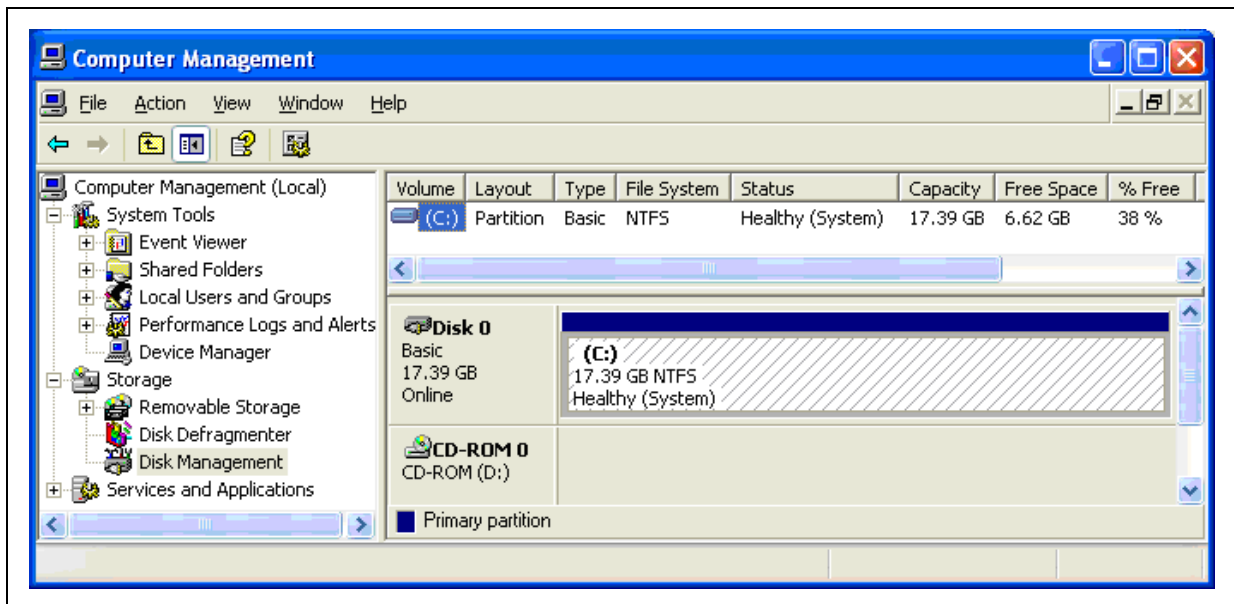




Figure 7. VM1 Disk Management view



Testing VM Independence

1. Download CPUBENCH V4.0.0.6 from <http://www.softpedia.com/progDownload/CPUBENCH-Download-1211.html> and install it to both VM1 and VM2 (and other VMs if you have started them).
2. Bring up the "Task Manger" using "Ctrl+Shift+Escape" to check the CPU Usage in the Performance panel.
3. Open "Date and Time Properties" by clicking Start->Settings->Control Panel->Date and Time.
4. Launch CPUBENCH 4.0.0.6, arrange the CPU Usage, the clock window and the CPUBENCH on the screen so you can see all three windows running.
5. Start the benchmark tests by clicking on the right icon of CPUBENCH window, record the start time with the clock running in the "Date and Time Properties". You will see 12 benchmarks running, one after another in the CPUBENCH Windows indicating the progress of the tests. Also you can see the CPU Usage in Windows Task Manager indicates 100% of the CPU usage when the benchmark is running.
6. When the "Graph of Overall_score" displayed in the "CPUBENCH V4.0.0.6 Compare" Window, the tests is complete, record the time. Record the total time to complete on VM1. It takes about 1 minute and 32 seconds.
7. Repeat step 2 - 6 to run the same tests on VM2. If you have started more than two Remote Machines, feel free to run all of them.
8. Once we have tested the benchmark on the individual virtual machine, we will start CPUBENCH running on both VM1 and VM2 (and any other VMs you have started) at the same time and check to see if the total time to complete all 12 benchmarks on the two (or more) systems is the same as running them individually.
9. The results are the same which imply the virtual machines are running independently on their core, there is no interference between the virtual machines even with 100 percent of CPU usages.



Conclusion

Many of the ideas and techniques used here will be transferable to future demo systems that can use VMM products specifically geared towards embedded and communications solutions. These types of products cater towards - and generally provide better performance for - I/O sharing, high I/O bandwidth, or real-time processing requirements that are more typical of these markets.

This paper can be used to set up and get familiar with virtual machines in an embedded-like environment, using hardware and software available now. From there, start a technical discussion with Intel representatives to understand if this type of solution can work as-is, or if a more tailored solution is required. Alternatively, many vendors in the Intel® Communications Alliance (via <http://www.intel.com>) will be able to provide more information about your options with respect to virtualized systems.

References

Title	Location
Intel® Virtualization Technology in Embedded and Communications Infrastructure Applications - Intel Technology Journal	http://www.intel.com/technology/itj/2006/v10i3/5-communications/2-intro.htm
Virtual Iron* User's Guide	http://www.virtualiron.com/fusetalk/forum/messageview.cfm?catid=16&threadid=74&enterthread=y
ConnectPort Display user's Guide	http://www.bb-elec.com/bb-elec/literature/ConnectUserGuide_a.pdf

Appendix A - Demonstrating Processor Affinity

Below is the screen capture of the domain info without processor affinity. You can see the domains are running on random CPU cores according to the scheduler assignment. Some comments are added to help understand which CPU the VM is running on.



```
[root@hdtme-linux1 support_tools]# ./testagent --exec="/bin/dominfo -c -f" 192.168.1.229
Status: 200 Started - PID 6945
Hypervisor Memory Info: total pages 1047064 (4090MB), free pages: 168348 (657MB), scrub pages: 0 (0MB)
Domain CPUs Online Memory(MB) Pages CPU Time State
0 8 8 16777215 114688 3099.830779652 .....R
0: 0 0x00000001 ob. 1978.994418962
1: 1 0x00000002 o,r 581.231001335
2: 2 0x00000004 ob. 11.023813143
3: 3 0x00000008 ob. 71.600350565
4: 4 0x00000010 ob. 8.660125874
5: 5 0x00000020 ob. 267.315602561
6: 6 0x00000040 ob. 157.509201157
7: 7 0x00000080 ob. 23.496297661
1 1 1 290 74201 42.028115648 .....R
0: 7 0xffffffff o,r 42.028124409 //VS1 is running on CPU 7
2 1 1 290 74201 36.144935731 .....R
0: 4 0xffffffff ob. 36.144936822 //VS2 is running on CPU 4
3 1 1 572 146352 32.836334589 .....R
0: 3 0xffffffff o,r 32.836342080 //VS3 is running on CPU 3
4 1 1 572 146352 58.493705979 .....R
0: 7 0xffffffff ob. 58.493729620 //VS4 is running on CPU 7
5 1 1 572 146352 30.060662259 ....B.
0: 1 0xffffffff ob. 30.060662259 //VS5 is running on CPU 1
6 1 1 290 74201 32.242924490 ....B.
0: 3 0xffffffff ob. 32.242924490 //VS6 is running on CPU 3
7 1 1 290 74201 29.248034059 ....B.
0: 7 0xffffffff ob. 29.248034059 //VS7 is running on CPU 7
[root@hdtme-linux1 support_tools]#
```

Following is another screen capture showing the domain info when processors are pinned. Each domain is now running on the specific processor core that was specified in xml file.



```
[root@hdtme-linux1 support_tools]# ./testagent --exec="/bin/duinfo -c -f 192.168.1.229
Status: 200 Started - PID 10913
Hypervisor Memory Info: total pages 1047064 (4090MB), free pages: 168350 (657MB), scrub pages: 0 (0MB)
Domain CPUs Online Memory(MB) Pages CPU Time State
0 8 8 16777215 114688 4266.828716838 .....R
0: 0 0x00000001 ob. 2788.000151893
1: 1 0x00000002 ob. 872.287094909
2: 2 0x00000004 ob. 16.585096107
3: 3 0x00000008 ob. 82.316379386
4: 4 0x00000010 ob. 21.891790351
5: 5 0x00000020 o.r 268.863577579
6: 6 0x00000040 ob. 189.153849002
7: 7 0x00000080 ob. 27.730830465
1 1 1 290 74201 21.134117597 ....B. //VS1 is running on CPU 1
0: 1 0x00000002 ob. 21.134117597
2 1 1 290 74201 16.185018197 ....B. //VS2 is running on CPU 2
0: 2 0x00000004 ob. 16.185018197
3 1 1 572 146352 14.889000829 ....B. //VS3 is running on CPU 3
0: 3 0x00000008 ob. 14.889000829
4 1 1 572 146352 21.531322321 ....B. //VS4 is running on CPU 4
0: 4 0x00000010 ob. 21.531322321
5 1 1 572 146352 12.029307228 ....B. //VS5 is running on CPU 5
0: 5 0x00000020 ob. 12.029307228
6 1 1 290 74201 0.664992690 ....B. //VS6 is running on CPU 6
0: 6 0x00000040 ob. 0.664992690
7 1 1 290 74201 23.634690544 ....B. //VS7 is running on CPU 7
0: 7 0x00000080 ob. 23.634690544
[root@hdtme-linux1 support_tools]#
```

