# AC 2007-904: A VIRTUAL MACHINE ENVIRONMENT FOR REAL-TIME SYSTEMS LABORATORIES

**Mukul Shirvaikar, University of Texas-Tyler**

MUKUL SHIRVAIKAR received the Ph.D. degree in Electrical and Computer Engineering from the University of Tennessee in 1993. He is currently an Associate Professor of Electrical Engineering at the University of Texas at Tyler. He has also held positions at Texas Instruments and the University of West Florida. His research interests include real-time imaging, embedded systems, pattern recognition, and dual-core processor architectures. At the University of Texas he has started a new real-time systems lab using dual-core processor technology. He is also the principal investigator for the "Back-To-Basics" project aimed at engineering student retention.

**Nikhil Satyala, University of Texas-Tyler**

NIKHIL SATYALA received the Bachelors degree in Electronics and Communication Engineering from the Jawaharlal Nehru Technological University (JNTU), India in 2004. He is currently pursuing his Masters degree at the University of Texas at Tyler, while working as a research assistant. His research interests include embedded systems, dual-core processor architectures and microprocessors.

# A Virtual Machine Environment for Real Time Systems Laboratories


## Abstract

The goal of this project was to build a superior environment for a real time system laboratory that would allow users to run Windows and Linux embedded application development tools concurrently *on a single computer*. These requirements were dictated by real-time system applications which are increasingly being implemented on asymmetric dual-core processors running different operating systems. A real time systems laboratory curriculum based on dual-core architectures has been presented in this forum in the past.[2] It was designed for a senior elective course in real time systems at the University of Texas at Tyler that combines lectures along with an integrated lab. The students are required to have at least one course in structured programming, and a course or prior experience with the operation of microprocessors, but Linux experience is not required. The lab procedures that were implemented include - running audio processing applications, building a Linux kernel, building an audio player application using cross compiler tools, testing a finite impulse response (FIR) filter and running a web hosting application. Instruction and application development on such architectures can be a major challenge involving one set of tools and hardware for each operating system. The common solutions are to use a dual-boot computer running Linux and Windows or use two separate computers running Windows and Linux respectively, neither of which is ideal.

Virtual Machine environments are becoming increasingly popular due to the various advantages they offer, especially in settings that involve development on multiple operating systems. Desktop virtualization software can be used to run multiple operating systems simultaneously on a single personal computer. A virtual machine is nothing but a single file or image embedded with the entire hardware configuration, operating systems and tools of a computing machine. The software allows Windows, Linux or Solaris to be run on networked virtual machines without the requirement of rebooting the system or partitioning the hard drive.[1]

This paper presents the results of implementing real-time systems laboratory experiments in a virtual environment. The VMware Workstation Edition package was used with Windows host operating system and Linux as the guest. The virtual machine implementation offered a wide range of benefits when compared to individually operating machines. The virtualization greatly improved the hardware utilization in the laboratory resulting in cost benefits. The major advantages offered by virtualization software are: simultaneous access to a multiple operating systems, automation of test sequences without multiple system reboots, migration of control between operating systems without user disruption, ability to create a preconfigured library of virtual machines thereby reducing setup time, and the ability to preserve the host machine content by isolating each virtual machine.[3]


## Introduction

Virtualization technology enables multiple operating systems to run concurrently on a personal computer. Virtualization is a means for providing the same system hardware to different operating systems running at the same time. Multiple operating environments can be run on the

same desktop without rebooting or repartitioning. The base operating system (OS) which hosts other operating systems is called the 'host operating system'. Any other operating system that can run on the host system hardware can be installed and run on the host and is called a 'guest operating system'.

All the guest operating systems are controlled by a software layer called 'Virtual Machine Manager' (VMM) or 'hypervisor' that takes complete control of the machine's hardware. A VMM (Figure 1) is a layer of software between the physical system's hardware and the operating system. It is primarily used to emulate the hardware resources of the machine. VMMs build on the lower level hardware platform and provide an interface to the higher-level software.[4] The VMM adds functionality below the existing operating system and application software. A virtual machine is nothing but an abstraction created by the VMM. There exist a variety of ways in which virtualization can be implemented. The basic form is known as 'full virtualization'. In this form of virtualization, the hypervisor acts like a fully emulated machine in which a guest operating system can be installed and can be run as an independent development environment in a fully functional mode. The VMM located between the guest operating system and the system hardware controls the CPU utilization by the guest OS, memory, storage and transfer of control from one virtual machine to another. The VMM isolates and protects the applications and data that are running in the operating environments and also acts like an interface between operating systems for networking and file sharing. This approach provides the flexibility to run a RISC based operating system as a guest operating system on an Intel-based host system.

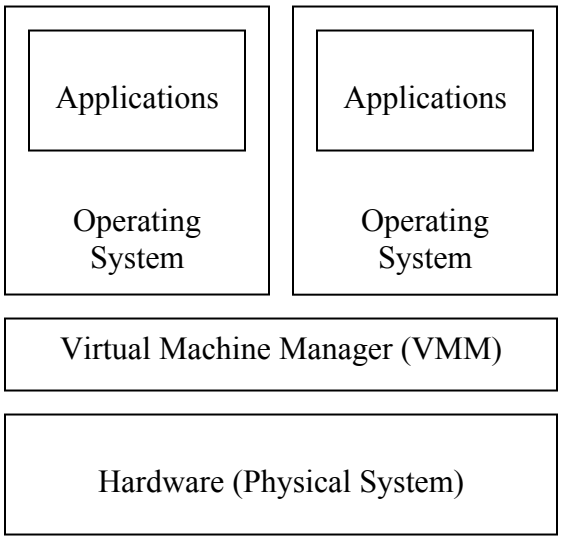



**Figure 1.** Virtual Machine Manager (VMM)

A virtual-machine platform creates virtual images of operating systems and makes guest operating systems appear and behave as completely independent environments running on the machine's hardware. A virtual machine can be treated as an independent piece of software because the entire computer system gets embedded into a software package and behaves like an application. This feature of the virtual machines allows virtual machine managers to host multiple virtual computers on a single physical computer, and also to control the resources

assigned to each individual system. Each virtual machine runs on the host system just like any other application software running on it. It may have the capability of using Windows or Linux as the host operating system and any other operating system as the guest system on it. An important advantage of virtualization software is that it can be installed like a user-level application program without requiring any special hardware support.

**Virtualization and Virtual Machines**

Virtualization of a system or component at a certain level of abstraction is achieved by mapping its interface and all the visible resources onto the interface and the resources of the underlying system. This is accomplished by using various resource management techniques like partitioning, simulation, and emulation. The virtualization layer normalizes the hardware of the physical system. The virtual machines installed on top of the virtualization layer behave as if they are installed on the same hardware, called the virtualized hardware. Virtual machine management systems actually encapsulate the system's hardware, operating systems and application programs into a single package.[5] VMMs are classified into Type I, Type II and hybrid types depending upon the platform on which they are built. The VMMs like VMware's ESX Server and IBM's VM/370 that are implemented directly on the hardware of the physical machine belong to Type I and VMMs like UMLinux and SimOS that are implemented directly on the host operating system are classified as Type II.[4] VMMs like the VMware's Workstation which operate on the physical hardware of the system but use the host operating system to perform I/O operations are the hybrid types.

The hypervisor or virtual machine monitor abstracts the computer's hardware and acts as a supervisor in providing security and tidiness of the shared resources. It provides the illusion of standard PC hardware within a virtual machine. All the virtual machines are unaffected by the changes in the hardware as they are completely shielded by the virtualization layer. The virtual machines created by most of the virtual machine development software consist of their own virtual drives, network interface cards, processors and other resources. VMMs are affected by many aspects of virtualization which decelerate them in the process of providing an interface similar to the hardware of the physical machine. In some systems these aspects depend upon whether the CPU is running and so they consume some time before getting activated. In order to accelerate the virtualization processes the guest tools may provide special drivers for the guest operating system. The guest operating systems of Type II VMMs require additional modifications after installation for the purpose of providing tools that accelerate the virtualization process.

**Benefits of Virtualization**

The virtualization of a system or a component of a system provides a lot of advantages during the real time implementation. The key benefits of virtualization are encapsulation, partitioning, and isolation.[6]

- **Encapsulation** provides a completely virtualized set of hardware to the VMs through which compatibility for most of the applications can be achieved easily. Virtual machines

are encapsulated into files, making it possible to rapidly copy and move a virtual machine.

- **Partitioning** of the virtual machine into disks or files provides a consolidated architechture which helps in allocation of resources to the VM in a controlled manner. This feature helps in running multiple operating systems like Windows, Linux, and NetWare and so on simultaneously on a single physical machine.
- **Isolation** provided to the VMs by the Virtualization software such as VMware builds up a secured environment where the data in one virtual machine cannot be accessed by another virtual machine but can only be obtained through secured network connections. The isolation also provides security to the host system by protecting it from VM crashes.

Other benefits that accrue from the virtualization approach are:

- The user can avoid the installation and maintenance of separate instances of complete operating systems. This reduces the workload and cost of operation.
- Inside virtual operating system instances, tasks can be initiated or terminated rapidly and it does not require a reboot of the entire operating system.
- The use of additional mechanisms to isolate the workloads allows the resource management tools to be used to precisely allocate resources to virtual operating system instances.
- The interface provided by the VMMs is convenient for adding much additional functionality such as primary backup replication and installation of tools for fault corrections.

Virtualization can provide a low-level kernel development environment through which technology can be introduced in a secure manner to students working on operating systems. The security here is provided by isolating the fatal problems like system crashes and other faults into a single virtual machine which allows students to work in secured and portable lab environment. This feature is of particular importance in a teaching laboratory environment.

**Past Work**

The IBM VM/370 was one of the first virtual machines developed in the 1960s. In the same period many other companies adapted the virtualization technology and started developing various kinds of virtual machines. VMware was first developed with an intention to bring virtual machine technology to industry-standard computers. The first Vmware product was Workstation released in 1999 and resulted from research on operating systems at Stanford University. During the last few years, many universities have started using VMware as an introductory operating-systems training course with a goal to provide the practical experience of working with multiple operating systems to the students.

Columbia University has been teaching the user-level Nachos simulator for introducing virtual machine technology to students.[7] The school of Computing at Christchurch Polytechnic Institute of Technology (CPIT), NZ has been using VMware to teach Microsoft, Linux and various other operating systems.[8] Their projects included working on multiple servers in multiple networks through a single physical system.

The concept of virtualization has been widely accepted by software giants like Microsoft and they initially used it for running applications on Windows operating systems, while simultaneously providing their users with access to their older systems. This gave them the advantage of making more efficient use of the hardware. This paved an easier way to operate applications in a virtual environment that is maintained in a secure and stable operating system. Eventually other firms like IBM, Hewlett Packard and Intel also turned towards virtual machine development software to work on multiple operating systems in multiple server environments.

**Technical Background**

The VMware Workstation software provides a virtualization layer that dynamically allocates the resources of a physical system to build a virtual machine. The workstation creates fully isolated and secure virtual machines that encapsulate an operating system and its applications. The virtualization is fully equivalent to a standard x86 machine.[9] The software can be installed on the host operating system and provides a complete virtual machine building environment. The features inside the VMware Workstation allow any application that runs on a standard PC to run inside a virtual machine with access to a full set of networking resources and devices. Each virtual machine therefore has its own CPU, memory, I/O devices, network interface cards, etc. Any application supported by the guest operating system works by utilizing these resources and devices.

VMware's Workstation software also creates disk partitions for each virtual machine called virtual disks that are stored as files on the host operating system.[21] The most advantageous feature is its ability to encapsulate the entire system. This feature not only provides quick access to the partitions but also makes the backing up and movement of files easier. The entire disk partition is saved as a single file and exists on the system as an independent file. The modifications made inside a virtual machine do not affect the other applications running on the computer. VMware Workstation uses a design called the Hosted Virtual Machine Architecture to virtualize I/O devices. The architecture of this design has the advantage of using an existing operating system for I/O device support and achieves almost the same performance figures as the native system.

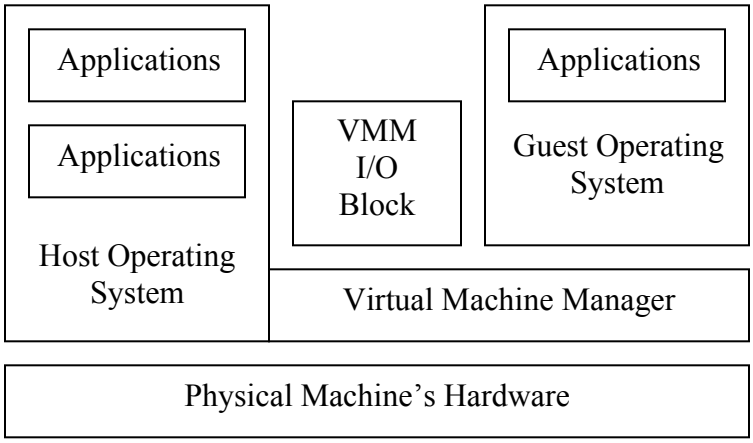



**Figure 2.** Structure of a Virtual Machine

Figure 2 shows the structure of a virtual machine in the hosted architecture. A driver called the VMDriver is loaded at startup into the host operating system by the application portion of the VM. Then the virtual machine monitor component is established which runs directly on the hardware. After that the real physical processor starts executing the VMM. The VMDriver controls the transfer from the virtual machine to the host machine and vice versa. The Workstation establishes a switch called world switch between the VMM and the host world. This switch saves all the user and system process states of the CPU. All the regular operations performed by the guest operating system run just like normal applications on the host system. The I/O operations performed by the guest operating system are controlled by the virtual machine manager. The VMM intercepts an I/O operation and switches it to the host operating system but it does not try to access hardware directly. After the operation is directed into the host system, it is carried out by appropriate system calls.

VMware uses a direct I/O architecture and establishes drivers into the hypervisor for high-performance I/O devices. The Service Console domain is used for devices that are not performance-critical. The hypervisor is protected from driver faults by using techniques like the private memory heaps for each driver that is used in the VM.

**Lab Environment Setup**

The primary goal of the labs is to introduce a stable, secure and portable virtual environment setup to students. The labs help them to get familiar with various operating systems like Windows XP and Linux. In the basic setup of the labs, Windows XP hosts Centos Linux as the guest through VMware's Workstation Edition 5.0.

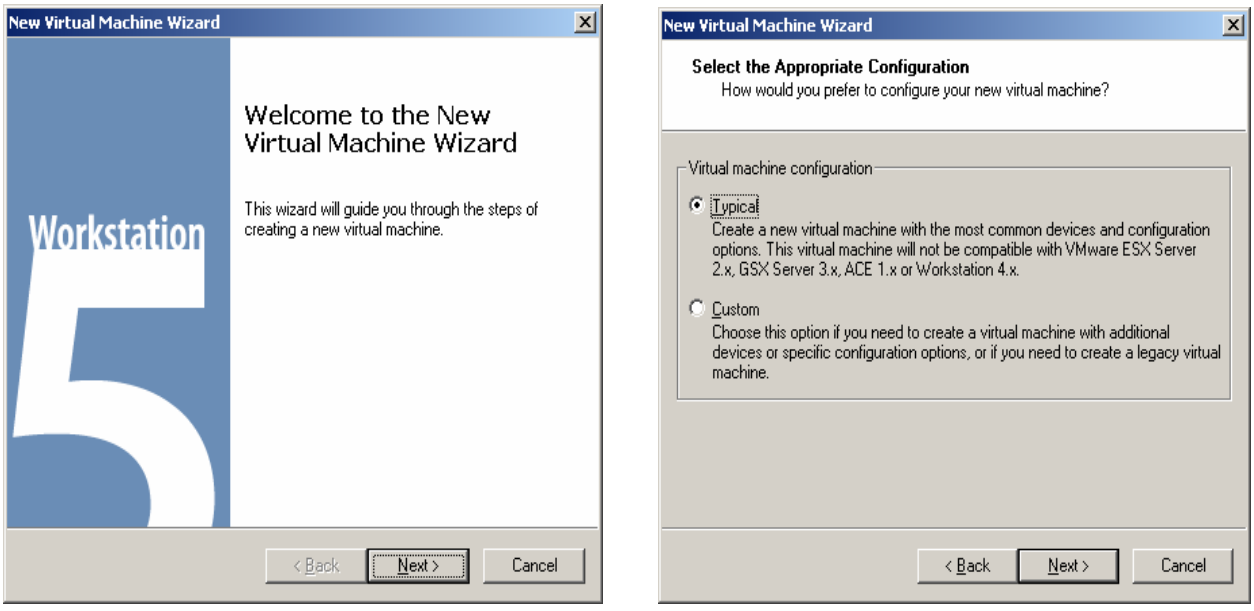



**Figure 3.** (a)Virtual Machine Wizard (b)VM configuration options

The laboratory curriculum starts with the installation of VMware Workstation and installing Centos Linux as the guest operating system. The process of installation includes various steps like partitioning of the disks, allocation of memory space and setting up network configuration for the virtual machines. The Workstation installation process is followed by the process of creating virtual images of the guest operating system and then setting up hardware device configurations needed for the projects.[6]

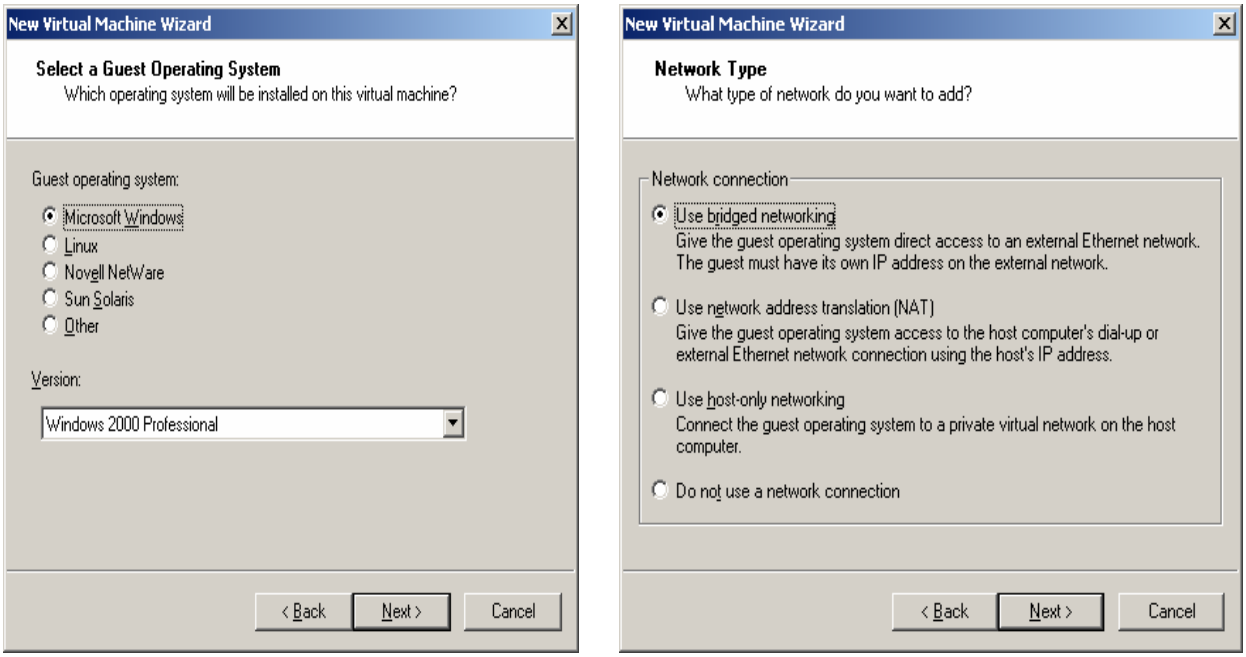



**Figure 4.** (a) Guest operating system options (b) Network type selection

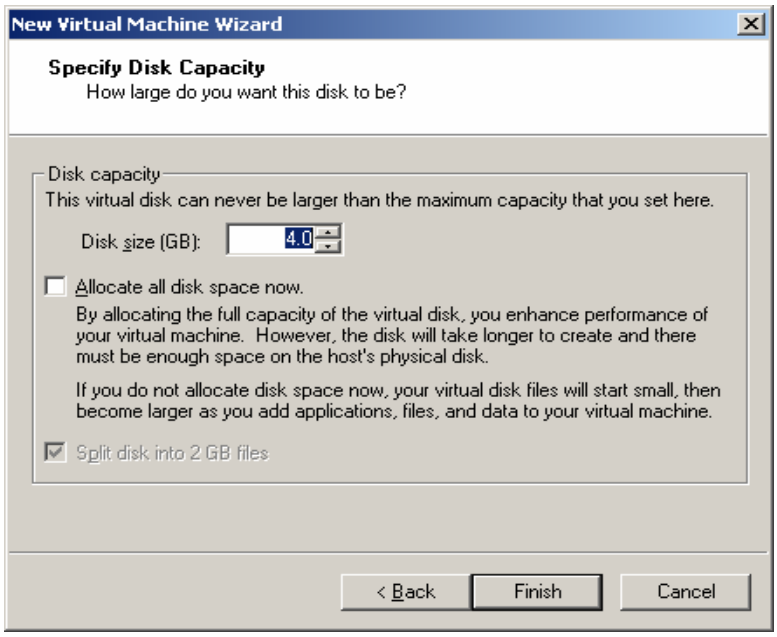
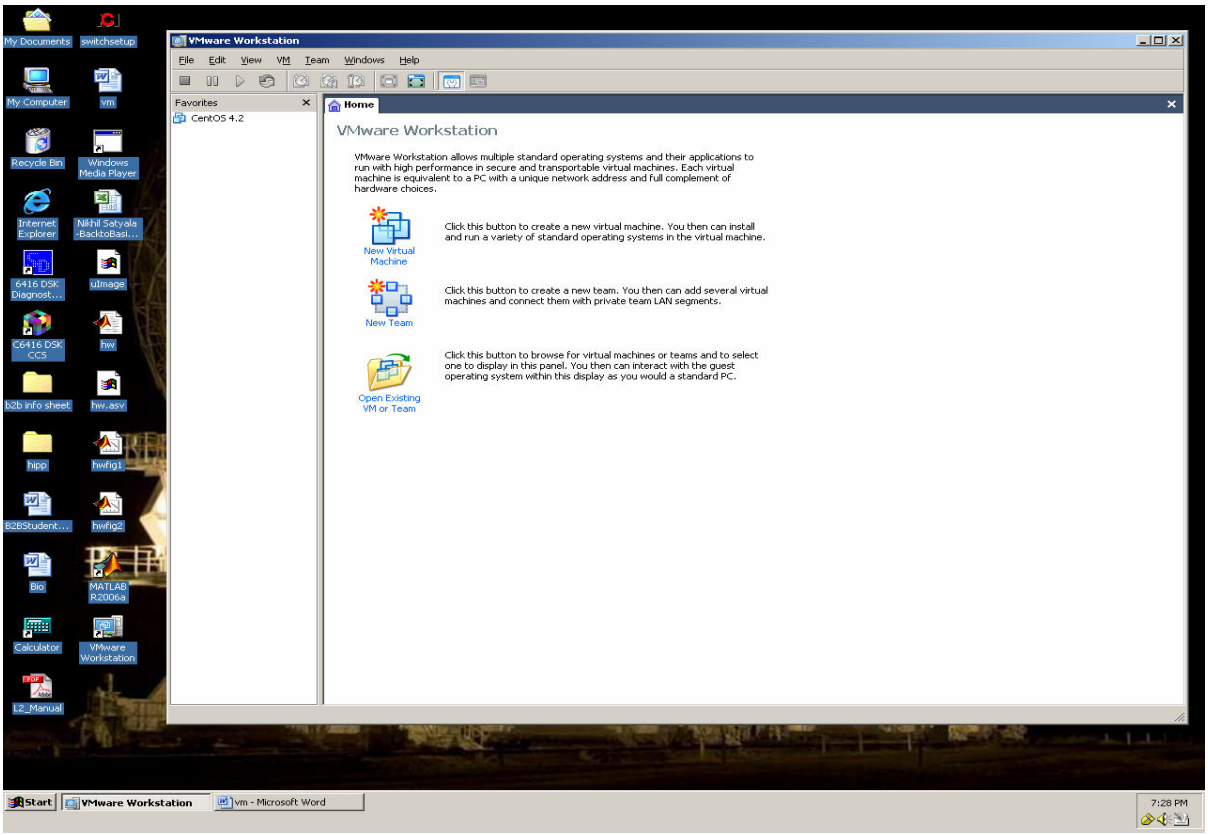



**Figure 5.** Virtual Disk size selection



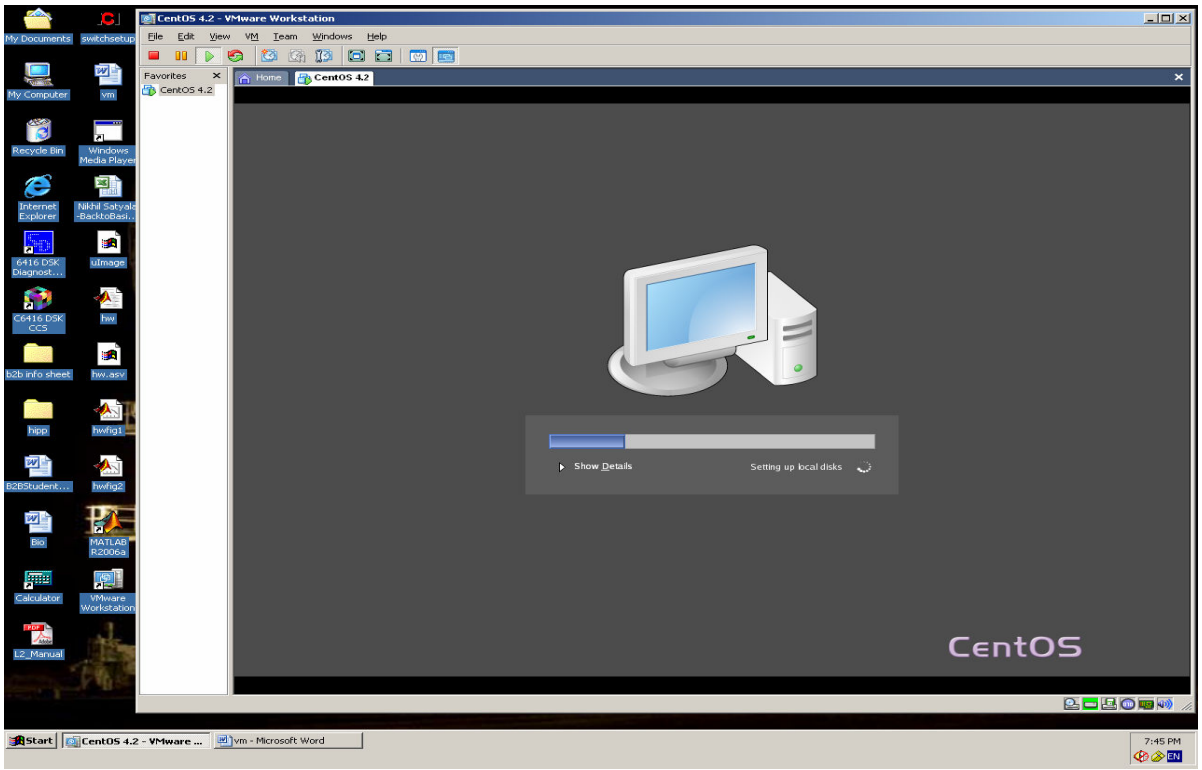




**Figure 6.** (a)VMware on Windows Desktop (top) (b)Installing guest operating system (bottom)

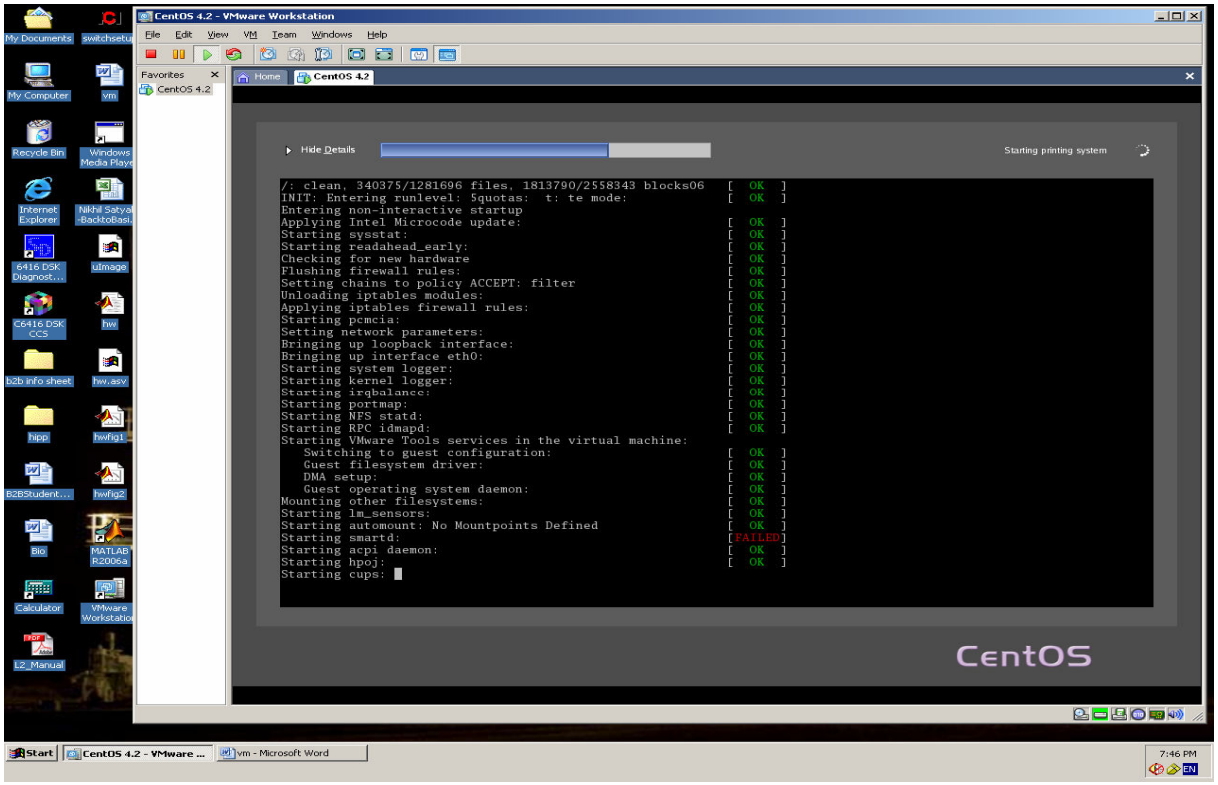




**Figure 7.** (a)Host and guest operating systems on the same desktop (top) (b)Linux login screen (bottom)

The VMware Workstation virtual machine wizard starts with a user friendly screen (shown in figure 3) that allows the user to select the appropriate configuration for the virtual machine. Selecting the 'typical' VM configuration option creates a VM that works with most common devices and the 'custom' option allows the user to opt for additional hardware devices. The wizard then gives the options for selecting a guest operating system and its version that will be installed on the virtual machine. The network type can also be selected during the procedure as shown in figure 4(b). Bridged networking, network translation or a private virtual network on the host computer can be selected and the virtual machine will be created as per the selected network configuration. The last step in the virtual machine creation is to select the size of the virtual disk depending upon the space available in the host system. The virtual machine development software will consider the selected size as the maximum size of the virtual disk as shown in Figure 5. The final step then creates a new virtual disk with the appropriate disk size. The virtual machine can be opened in the VMware Workstation window as shown in fig 6(a). The selected guest operating system can be installed on it as shown in fig. 6(b).The figures 7(a) and 7(b) show the running of the host and guest operating systems at the same instant and on the same desktop. The configuration settings chosen for our implementation were 10GB of disk space, serial port interface and bridged networking.

**Real-Time Lab Curriculum Implementation**

A real time systems laboratory curriculum based on dual-core architectures has been presented in this forum in the past.[2] It was designed for a senior elective course EENG 4325 Real Time Systems at the University of Texas at Tyler that combines lectures along with an integrated lab. The students are required to have at least one course in structured programming, and a course or prior experience with the operation of microprocessors, but Linux experience is not required. The curriculum aims towards educating students in the real-time systems environment. The basic lab projects [3] were designed on the Open Multimedia Application Platform 5912 (OMAP 5912) starter kit module developed by Texas Instruments (TI). The OMAP platform is based on dual-core processor architecture and consists of an ARM processor and C55x DSP.

The labs include procedures [20] to get familiar with

- basic Linux commands and platform development applications
- Linux Development Environment (LDE)
- real-time audio processing with the help of digital signal processors
- real-time image processing techniques and
- web hosting techniques

The labs use Linux on the ARM core and the DSP core utilizes the DSP-BIOS Kernel provided by TI. All the operations performed on the OMAP starter kit are controlled from the Linux operating system's command line that exists virtually on the physical system. Thus all the communication between the virtual machine and the ARM and DSP chips is carried out with the host system as the mediator. The host system helps the virtual machine to access all the hardware required to communicate with the processors residing on the kit.

**Lab 1 – Development Platform and Applications**

The first lab is an introduction to development with the OMAP Starter Kit 5912 (OSK5912). A set of basic Linux commands are introduced and a procedure to establish a local file system used

for the development platform is implemented. A simple audio processing program is executed to demonstrate the utility of the dual core OMAP 5912 processor. An audio signal is filtered through the AIC23 codec using the C55x DSP in the OMAP5912. The inter-processor communication between the ARM processor running Linux and the audio application running on the DSP is highlighted through the demo program. This process involves changing the filter parameters in real-time and requires serial communication to carry out the control commands from the operating system to the digital signal processor.

**Lab 2 – Linux Development Environment**
This lab introduces a procedure to build a Linux kernel to be used by the OMAP5912 processor. The bootable kernel is then burnt on to the OSK5912 flash memory. The procedure also includes the process of establishing a Network File System (NFS) mountable partition on the Linux host machine. The transfer of image from the virtual machine to the ARM processor is again carried out with the help of the host machine to access the serial communication port. Finally a test application program is built and run to verify system operation of the kernel.

**Lab 3 – Real-Time Application Installation**
This lab discusses the process involved in the installation of an application on the newly built kernel on the OSK5912 flash memory. The lab uses the open source community to acquire an audio application and the required compression libraries for application support. After acquiring the necessary files, the cross compiler tools for the OMAP 5912 Starter Kit (OSK 5912) are used to build the audio player application. The process requires access to the internet to download and install application software. The procedure thus demonstrates the access and utilization of network devices by the virtual machine.

**Lab 4 – Real-Time Imaging Techniques**
This lab demonstrates real-time imaging techniques to alter images and exchange formats. The image compression and manipulation utilities are acquired from the open source community. The procedure also includes various commands for a user logged onto the board to alter the file system by making calls to the board. All the calls are hence generated inside the virtual OS and are carried out to the board with the help of the host OS.

**Lab 5 – Introduction to Web-Hosting Techniques**
This lab deals with a procedure that introduces the web hosting features of the OSK 5912 board and of the operating system. A Common Gateway Interface (CGI) script and a simple HTML page are designed to be hosted by the board. The file system for the board contains a directory that can host web pages that can be served over the connected network.

**Lab 6 – Introduction to Dual-Core Development**
Dual-Core processor development tools are introduced through this lab by building and running applications that can utilize both processing cores available with the OMAP5912 processor. The Linux DSP tools are used for compiling the DSP source code. The controlling of the DSP processor is carried through the Linux running on the ARM processor core. This lab demonstrated the ARM/DSP functionality of the OMAP5912.

**Lab 7 – FIR Filtering**

The lab discusses the implementation of a Finite Impulse Response (FIR) filtering application that can either accentuate or attenuate a selected band of frequencies of an audio signal. Inter process communication is the mostly highlighted part of this lab. The audio file is made to communicate with the appropriate DSP task and a thread is created for the user to control the filter coefficients. Other tasks include reading blocks of data simultaneously to transfer to the DSP and sending the data to the audio codec for sound input after processing.

The labs provide to the students a thorough exposure to the concepts of a FIR filter, inter-processor communication, and other applications in a real-time environment. Thus the lab curriculum deals with a wide variety of topics that provide the practical experience of working with virtual machine technology. All the laboratory procedures can be accessed at the website http://ee.uttyler.edu/omap/labs/labs.htm [20]

The communication between the OMAP5912 board and the operating system is initiated and controlled by a serial communication program like "minicom". As the virtual machine is completely shielded from the physical systems hardware the serial port on the host machine cannot be accessed directly by the virtual machine. In this case the virtual machine monitor (VMM) takes control of the process, interfaces with the host operating system and requests for the serial port access.

**Implementation Issues and Resolution**

The implementation of the labs in the virtual environment resulted in some issues that had to be resolved, mainly related to hardware accessibility.

Serial Port Issue: The VMware Workstation virtual machines interact with the hardware of the system through the host operating system. The host operating system takes control of all the communication between the board and the virtual machine. When a serial communication program like minicom is initiated on the virtual machine or the guest OS to communicate with the OMAP5912 board, the host operating system uses a serial port which is not currently connected to any other OS or virtual machine.

VMware Workstation was built with four serial ports with virtual 16550A UARTs. The virtual machine manager maps these ports to the physical ports of the host system. These ports can be connected to actual devices or can be used to access or perform operations on the files of the host system. The four ports COM1 – COM4 share only two Interrupt Requests (IRQs) namely IRQ 3 and IRQ 4. Some versions of Linux do not allow sharing of IRQs. In such a case the access to COM3 and COM4 ports may not be possible.

Bridged Networking: The most convenient and easiest way to provide access to the virtual machine is to connect the system to an Ethernet network. Selection of the bridged network option while creating the virtual machine provides it with the configuration necessary to get connected to the Ethernet network. In this type of network configuration the host acts like a network bridge. The guest operating systems are provided with access to the local LAN through the host system. Usually up to three fully bridged networks can be established on a virtual machine.

In a bridged network each virtual machine is supposed to have a unique IP address. The virtual machine can hence become a full participant in the network and can have the privileges to communicate directly with other machines on the network just like a physical computer. Using the same network address to all the virtual machines running on the system will lead to problems like miscommunication when working in a networked environment. The virtual machines can acquire their IP addresses from DHCP server or they can also be set directly by the user.

**Virtual Environment Performance**

The performance of the workstation depends on various factors like the number of virtual machines that are currently running, memory available on the host system, host OS disk fragmentation and others. For the labs in this project, various features of the VMM helped in delivering a good performance.

*Multiple snapshots:* The multiple snapshot facility allows the user to capture the state of the virtual machine at different points in procedures. This functionality helped in accelerating testing and debugging of the labs. The snap shot feature allows the user to easily switch between configurations by providing a thumbnail display of the shots.

*Multiple Virtual Machines:* During the lab procedures, no reliability issues were recorded while running multiple machines. In addition, the lab experiments showed the same type of performance as in a real machine. There were no significant time delays during the execution of the procedures, except intermittent minor delays in the graphical user interface response time.

*Cloned Virtual Machines:* The software can create copies of the existing virtual machines with links to the original machine or as a complete standalone copy of the machine. This is very useful when the developer has to perform a task that will drastically effect the configuration of the system.

*Networking:* The implementation of the labs based on web hosting and real-time application installation need a highly reliable network to download binaries for the applications and to launch a secured web page with board as a host. The bridged networking between the network and the guest system provided a reliable network service.

*Secure Environment:* Virtualization enabled a highly secure environment by isolating each virtual machine. At the same time, it provided the user full independence to work on the guest operating system. Any fatal errors or system crashes do not affect other virtual machines. Recovery is possible in a short time by removing the virtual machine as a whole.

A number of tests were performed to compare the performance of the virtual environment with a real Linux machine. The physical machine used for the tests was a Dell Optiplex desktop system with an Intel® Pentium® 4 processor with a clock speed of 3.00 GHz and 1GB of RAM. VMWare was running with 256MB of RAM, which is much greater than the recommended minimum memory of 32MB for our configuration. The virtual machine has a maximum size of 12GB for the virtual drive. Our Host operating system was Microsoft Windows XP Professional SP2. CentOS Linux was running as guest operating system. Table 1 shows a performance comparison between a virtual machine and a real system, measuring critical timings for the following operations:

- Boot up time (OS): For a real machine this is the time taken from the moment it is powered on till the machine is fully booted. For a virtual machine this is the time taken

for the virtual machine software (like VMware) to start up on the host machine plus the time taken for the virtual machine to get completely booted up after it is powered on.

- Serial port speed: This test was performed during the second real-time lab in the curriculum where a 776 Kb file was to be transferred onto the OMAP5912 board using a serial communication port. The connection between the board and the virtual machine (CentOS Linux) was established using the serial communication program 'minicom'. The time taken for the file transfer depends upon the guest operating system and its capability to transfer data. It may be different for different guest operating systems.
- Network speed: The file download time actually depends on the speed of the network and the size of the file that is being downloaded. The virtual machine was slightly slower on the same network as the real machine. The VM performance on the network was observed by downloading a 5MB file through a 100Mbps Ethernet network to the real machine and the virtual machine. The results showed that the VM running in bridged networking mode was only 9% slower than the native machine.

**Table 1.** A performance comparison between a VM and a physical machine

| PERFORMANCE FACTOR | VIRTUAL MACHINE | REAL MACHINE |
|---|---|---|
| Boot-up time (OS) | 1 min 10 sec | 25 sec |
| File transfer time through serial port. File size: 776 Kb | 2 min 21 sec | 2 min 10 sec |
| File download (Network: 100Mbps) (File size: 5MB) | 4 min 45 sec | 4 min 26 sec |

Additional tests were performed to measure the performance of a virtual machine in terms of overall user experience. It was observed that the response time or the latency for a mouse click or double click in the VM was greater than that on the physical machine. This lag is very minute and is not measurable with direct human observation. The average time taken for the VM to complete a set of simple tasks followed by a mouse click or double click was measured for comparison with the real machine. Similarly the response time for various applications to initialize is also comparatively higher on the VM. The application response time was measured by opening and running some of the most frequently used applications multiple times on the real and virtual machines. The total time taken was then averaged to determine the response time. Table 2 shows the average response time for common tasks performed on the virtual and real machines. The figures in the table indicate the latency or differences in response times for various applications. An overall comparison shows that the virtual machine takes more time to complete the processes. The difference is not significant and is well with the limits of user tolerance from a "qualitative" point of view. This is especially true if the user does not have prior exposure to a real machine, the evaluation of such user experiences being subjective in nature.

The other significant factor that affects the performance of the VM is the amount of memory that is allocated to it. A huge amount of memory should not be allocated to the VM as it results in slow performance by the host. Allocation of very small amount of memory results in the guest

performing slower. Compute-intensive applications running on the VM have less effect on the VM performance but if the application does a lot of network or I/O operations, the performance hit is higher.

**Table 2**. Comparison of response times for various tasks and applications

| TASK | VIRTUAL MACHINE (AVERAGE RESPONSE TIME) | REAL MACHINE (AVERAGE RESPONSE TIME) |
|---|---|---|
| Closing windows followed by a mouse click (averaged over 10 clicks) | 1.3 sec | 0.9 sec |
| Opening up serial communications program 'minicom' to connect to the OSK (averaged over 5 times) | 4.1 sec | 3 sec |
| Opening Totem media player (averaged over 5 times) | 1.85 sec | 1.5 sec |
| Opening a word processing application (averaged over 5 times) | 4 sec | 3.5 sec |
| Extracting a compressed file from the command line (file size: 5 MB) | 3 sec | 2 sec |
| Auto-run CD initial response (averaged over 5 times) | 6 sec | 3.5 |

**Host Vs Guest**

The host and the guest operating systems use the same system resources to run applications. It would provide tremendous insight into the system operation to observe how they share these system resources. The 'Veeam Monitor for VMware' [22] software was used to comparatively observe the resource usage, memory usage, disk usage, and network usage by the host and guest operating systems,. This software provides a graphical and numerical comparison between the machines by monitoring the applications running on them. Various test results for a comparison of CPU usage were obtained by providing the same workload to both the machines and then monitoring them using the Veeam monitor.

Figure 8 shows the CPU usage by the host machine and the virtual machine during the download of a 10 MB file on a 100 Mbps Ethernet network. The green line in the graph indicates the virtual machine and the white line indicates the host machines CPU usage. The graph clearly shows that the CPU usage by the virtual machine is much lesser than that of the host.

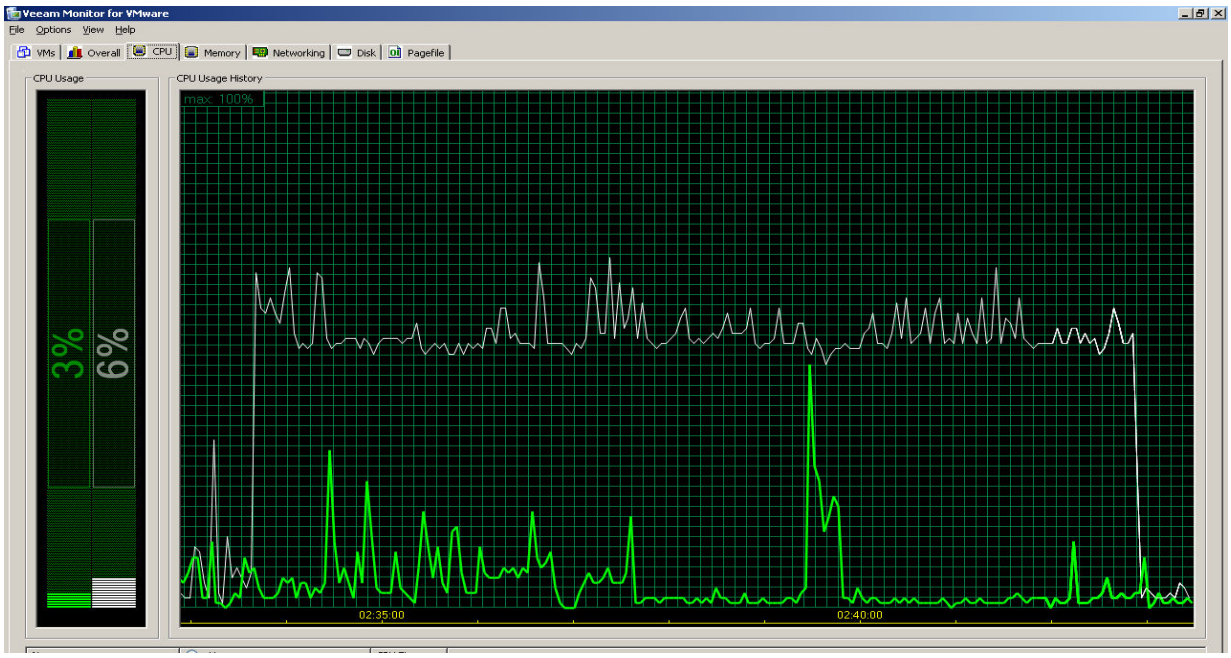



**Figure 8.** CPU usage during a file download: host (white line) and guest (green line) machines

The machines were also tested by running a common word processing application. The CPU usage by the virtual machine was found to be greater than that of the host machine on few occasions but not to the extent of obstructing host machine CPU usage. The graph in Figure 9 shows the dominance of the host machine except in the first quarter.

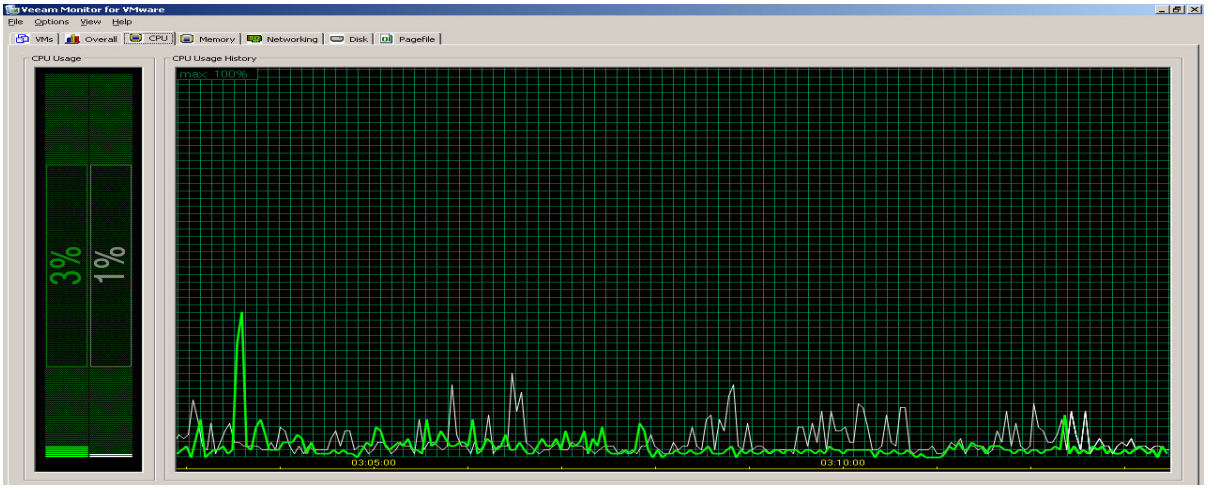



**Figure 9.** CPU usage while running a word processing application: host (white line) and guest (green line)

Figure 10 shows the CPU usage during a file transfer operation from the OS onto the OMAP5912 board. For this test the green line (guest) in the first half shows the CPU usage during the file transfer through the serial port using minicom and white line (host system) in the

second half of the graph shows the CPU usage during the same file transfer through the serial port using hyper terminal.

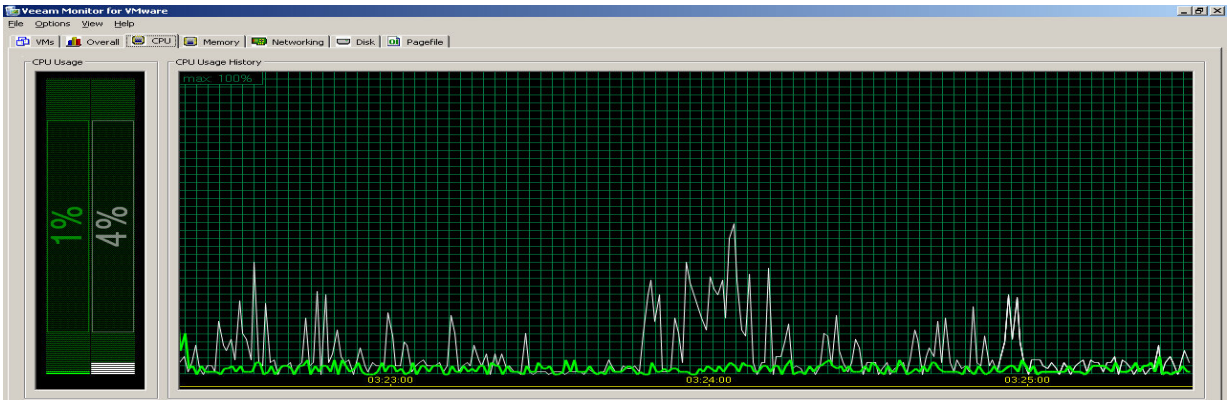



**Figure 10.** CPU usage during file transfer through serial port: host (white line) and guest (green line)

## Conclusion

The real time laboratory procedures were implemented and tested successfully in the virtual machine environment. VMware Workstation provided a stable base for the project, with reliable networking and hardware support as well as numerous special features. The result is a superior environment for a real time system laboratory that allows users to run Windows and Linux embedded application development tools *concurrently* and *on a single computer*. The performance of the virtual machine environment was extensively compared to that of a real machine using various metrics of interest, helping characterization from a quantitative "performance" and qualitative "user experience" viewpoint.

The virtual lab environment offers major advantages to educators. Every virtual machine creates an independent execution environment, and thereby saves the cost of maintaining additional hardware resources. Virtualization offers a significantly secure environment for the students to experiment and learn in, within the bounds of university information technology security rules. In addition, students are willing to think outside the box and experiment, since disaster recovery is as simple as removing and creating a new virtual machine.

## Bibliography

1. http://www.vmware.com/products/ws, "VMware website", 2006

2. Mark Humphries and Mukul Shirvaikar, "Real Time Systems Laboratory Development: Experiments Focusing on a Dual Core Processor", ASEE Conference, May 2006

3. www.vmware.com/pdf/dev_test.pdf, "Accelerate Software Development, Testing and Deployment", White paper – VM Ware, May 2006

4. Samuel T King, George W Dunlap, Peter M Chen, "Operating System Support for Virtual Machines", USENIX Technical Conference, 2003

5. Mendel Rosenblum, Tal Garfinkel, “Virtual Machine Monitors Current Technology and Future Trends”, IEEE Computer Society Publication, May 2005

6. “http://www.vmware.com/virtualization/”,“http://www.vmware.com/pdf/ws55_manual.pdf”, VMware Websites, 2006

7. Jason Nieh, Ozgur Can Leonard, “Examining VMware”, Dr. Dobb’s Journal, August 2000

8. Ricky Watson, Eddie Correia, “A Virtual Solution to a Real Problem: VMware in the Classroom”, White Paper, School of Computing, Christchurch Polytechnic Institute, NZ

9. “Virtualization: Architectural Considerations and Other Evaluation Criteria”, White Paper, VMware Inc., 2005, http://www.vmware.com/solutions/whitepapers.html

10. James E Smith and Ravi Nair, “The Architecture of Virtual Machines”, IEEE Computer Society Publication, May 2005

11. Keith Adams, Ole Agesen, “A Comparison of Software and Hardware Techniques for x86 Virtualization”, ASPLOS ’06, Sanjose, CA, USA, 2006

12. Peter M Chen, Brian D Noble, “When Virtual Is Better Than Real”, Department of Electrical Engineering and Computer Science, University of Michigan

13. Simon Shumate, “Implications of Virtualization for Image Deployment”, Dell Power Solutions, October 2004

14. Danielle Ruest, Nelson Ruest, “Virtualization from Concept to Reality”, White paper, Insight and Hewlett Packard, 2005

15. J.E. Smith, Ravi Nair, “An Overview of Virtual Machine Architectures”, White paper, Elsevier Science (USA), November 2005

16. Jeremy Sugerman, Ganesh Venkitachalam, Beng-Hong Lim, “Virtualizing I/O Devices on VMware’s Workstation Hosted Virtual Machine Monitor”, Proceedings of 2001 USENIX Annual Technical Conference, Boston, USA, June 2001

17. Harry Bulbrook, “Using Virtual Machines to Provide a Secure Teaching Lab Environment”, White Paper, Durham technical College, Durham, USA

18. Mendel Rosenblum, Tal Garfinkel, “A Virtual Machine Introspection Based Architecture for Intrusion Detection”, White Paper, Computer Science Department, Stanford University

19. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield, “Xen and the Art of Virtualization”, SOSP ’03, October 2003

20. http://ee.uttyler.edu/omap/labs/labs.htm, “Website for Real Time Lab Procedures based on OMAP 5912”

21. Steven Warren, “The VMware Workstation Handbook”, 2005 Edition

22. http://www.veeam.com/veeam_monitor.asp, Source for 'Veeam Monitor for VMware' software.