INTRODUCTION OF VIRTUALIZATION IN THE TEACHING OF OPERATING SYSTEMS FOR CS UNDERGRADUATE PROGRAM[*]

Kevin Grammer, Jack Stolerman, and Beifang Yi
Computer Science Department
Salem State University
352 Lafayette Street, Salem, MA 01970
(978) 542-7426
kevingrammer@gmail, viper11@gmail, and byi@salemstate.edu

ABSTRACT

For many colleges, virtualization is a low-cost solution for providing hands-on lab activities for computer science courses. This paper describes the design and implementation of a series of projects for an undergraduate operating systems course. By utilizing Linux virtual machines on students' personal computers, these projects teach reinforce students' understanding of operating system concepts as well as teach students the basics of virtualization. After completing the projects, students were surveyed about their experience and a summary of their responses is presented here.

INTRODUCTION

The study of operating system (OS) principles is an integral part of the computer science undergraduate curriculum. Beyond lectures and reading textbooks, hands-on activities add significantly to a student's understanding of computer science topics. The CS2008 Review Taskforce emphasizes that the study of operating systems should include a "laboratory component to enable students to experiment with operating systems" [1]. While some major universities are able to provide dedicated computer labs for this purpose, smaller colleges must use what they have at hand and often cannot provide labs for every course [4, 6]. The challenge for these colleges is to find other ways to provide students with high-quality, affordable hands-on experience [2, 3].

[*] Copyright © 2011 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

Virtualization offers a lightweight, low-cost, practical solution to this problem. The student installs a virtual machine on his or her personal computer and it runs on top of the computer's host operating system [12]. Separation between the host OS and the virtual machine allow the students to manipulate these virtual machines without the risk of damaging the host OS [4, 8].

In this paper we will discuss applying virtualization in teaching an undergraduate OS course with hands-on projects. Projects were designed to introduce students to the Linux operating system. The main objectives of the projects were to have students update the Linux kernel, add a system call, and work on process-and thread-related projects in the Linux environment. We have also included the comments and feedback we received from the students in this paper.

BACKGROUND

As personal computer speeds and storage space increase rapidly and OS virtualization becomes more easily accessible and commonly used professionally, colleges are more seriously considering adopting virtualization in the classroom. Some schools have already developed virtualization techniques to bring students more accessibility to lab-based activities for both university students and distance learning students [7]. Not only does it cut lab costs for colleges that use virtualization as a teaching tool, but it gives students necessary exposure to the latest real-world technologies. Most students' personal computers meet the minimum requirements for running at least one virtual machine, making it a practical alternative to using university computer labs. Many universities have already implemented virtualization to give their students hands-on experience in system-oriented computer science courses such as operating systems, system administration, and networks courses [4, 5, 7].

OS virtualization allows a CS professor to give his or her students exposure to different operating systems that they may not have prior experience with. Rather than simply learning the software of a single OS, using a variety of operating systems emphasizes the fundamental and lasting concepts of operating systems by revealing the commonalities in function between them [2, 10]. In addition, virtualization gives the professor flexibility as to which OS to use to teach each concept. For example, a professor might use Linux to help students understand the distinction between user and super-user security levels [7].

At Salem State University, Operating System Principles is a 3-credit hour undergraduate course. While many CS courses here have separate lecture and lab time, this course has no accompanying in-class lab hours. From our past experience in teaching OS, students often complained of the theoretical descriptions of OS topics and algorithms without lab practice to help solidify an understanding of the internal structure of the OS. Due to the limited resources in our department (students do not have administration rights on the machine) and course setting (no in-class lab time), we must find non-traditional solutions to overcome these challenges— to design OS projects that can be implemented on students' personal computers.

PROJECT GOALS AND PREPARATIONS

We set the goals of designing the OS projects as: (1) introduction to virtualization — the students will not only learn a new technology but also complete the OS projects without affecting their personal computer environments; (2) one installation is a Linux operating system — the students will get experience with different operating systems than the norm (Windows/Mac) and work with free, open source software (most CS students here have a very limited knowledge of Linux); (3) patching the kernel with a newer version — the students will acquire direct knowledge of compilation, configuration, and construction of Linux; and (4) adding a system call to the new kernel and practicing with the new kernel through a simple C program — the students will have a chance to look inside the OS to obtain a basic understanding of internal structure of one operating system and to reinforce OS principles.

To achieve these goals, we polled the students in an initial survey. Questions from the survey included inquiries about the students' experience with computers, their computer usage (such as which operating systems and programming languages they had used) and configurations of their personal computers (such as RAM/hard-drive capacity). From the feedback (of 25 replies at the beginning of 2010 Fall), we noticed the following:

- All the students own one or more computers: all students except one have laptops; about 40% have both desktop and laptop.
- Only two students (8% of total) have a computer with 1 GB RAM; 20% with 2GB RAM; 60% with 4GB RAM; 12% with 6 or 8GB RAM.
- 40% of students did not have any knowledge about Linux; 40% have very limited experience with Linux.
- 20% of students did not use any command line environment; 40% had limited experience through the using of Windows' DOS.
- More than 80% of students used *only* Windows operating systems.

Previously, we took a similar survey in the Fall of 2009 for the OS class and received very similar results indicating a little lower percentage in the student population that had computers with 2GB or more of RAM.

These statistics provided us with guidelines to design OS projects. First, we decided to use VMware Workstation as the virtualization software package for virtual machines. Our considerations were (1) compared with other virtualization solutions, VMware is more reliable, more widely adopted in industry, easier to set up, and provides more management tools [9] and (2) each student's personal computer meets the hardware requirements needed to install VMware. We also encouraged students who had experience with virtual machines to use alternative virtualization packages such as VirtualBox.

Next, we made use of the concepts and topics from the course textbook [11] for the implementation of the projects. Thus, students should be familiar with these topics from reading the textbook. We focused on the following points in designing the projects: (1) any student with no background in Linux should still be able to complete them, (2) students should be given a chance to "see" the inside of Linux and to make some modifications to it, and (3) some projects which covered the most important concepts of the OS (such as processes and threading) should be simple enough to be implemented in both C and Java programming languages (Java is the first and required programming language taught at Salem State University. Most students had little to no prior experience coding in C when taking the OS course).

PROJECT IMPLEMENTATION AND RESULTS

Project 0: Open Source OS and Virtualization

This project begins with a guide to downloading VMWare (or another virtual machine software) and installing the Ubuntu operating system. The Ubuntu file, although readily available for free download on the Internet, is also provided on the university network and distributed on CDs and thumb drives to students in class. Ubuntu is booted within the virtualization software and from here on, students work within the Ubuntu OS rather than the host OS.

After a basic introduction to the Ubuntu GUI and some of its preinstalled applications (Firefox, Open Office, Gimp, etc.), the student is given instructions to use the command line interface. The student tries some commands, such as *cp*, *mv*, and *uname* to get some information about the OS. Next the student makes a directory, changes directories, and lists parent files. Once the student is familiar with some basic commands, he or she is asked to create a *hello.cpp* file and enter some lines of code (provided in the project description). In order to compile and run the code as a C++ program, the student must first download and install the g++ software package from the command line. As the student completes each step of Project 0, he or she takes screenshots for submission.

The goal of Project 0 is to give the students some familiarity with the virtual machine, Linux, and the CLI. The project focuses on how to do a variety of basic operations with the Linux virtual machine, such as how to install a virtual machine running Linux, how to navigate Ubuntu's GUI, and how to get basic functionality from the CLI. With the completion of Project 0, the student not only has a bit of experience with Linux and the command line, but also has a "how to" reference for the next project.

Project 1: Updating the Linux Kernel and Adding a System Call

This project is a step-by-step guide to updating the Linux Kernel with a later version and adding a system call to the kernel. Much of what the student is asked to do in Project 1 is complex and most likely beyond his or her basic understanding of operating systems at the time when the project is assigned. Project 0 has given them some understanding of the Linux environment and command line to better follow along.

Initially, the student opens up the CLI and logs in as the root user. From here on, the student can write to the system¡¦s directories and must be careful not to accidentally change file or directory names. Because the student is doing this on a virtual machine, if they do make an error, they only run the risk of corrupting the virtual machine, and cannot directly affect their host operating system. Project 1 then takes the student through updating the Linux package and downloading the new kernel.

In order to add a system call to the kernel, the student must act as the root user and edit files within the kernel. Project 1 guides the student through each step of the process, editing the appropriate system files and adding a simple system call function to the kernel. In the case of Salem State University, most OS students have very little experience coding in C and C++. Project 0 gave students a chance to write and compile a C/C++ program. Project 1 requires the students to manipulate C files. The student should gain enough knowledge from Project 0 to successfully manipulate these files and at the same time become more familiar with the structure of C files.

After adding the system call, the student must configure the kernel using the kernel configuration menu. The student loads the new configuration and then builds the new kernel, appending his or her first initial and last name to the kernel version string. The compiling process takes anywhere from two to five hours. By witnessing the compiling, students said they gained an appreciation for how massive a task compilation really is. The student then installs the kernel and tests the newly added system call with a C test program. Although it is only a simple C program, it re-enforces the basics of C and its uses in the Linux kernel. Writing a working C program that makes a student-created system call can be a very powerful tool for convincing Java-based students that at least a basic understanding of the C programming language is important.

At the conclusion of the project, the student submits screenshots of the process to the professor. The greatest satisfaction the student gets is seeing his or her name appended to the kernel version name while getting a good look at the composition of the kernel of how it functions. The student also gains valuable experience utilizing the CLI and gains an appreciation for the root user power of the command line.

Project 2 and Project 3: Processes and Multithreading in C/Java

Project 2 builds on the basic C concepts learned from the previous two projects. In this project, students must create multiple processes that communicate with each other using pipes. A message is sent from a parent process to a child process. The message is then modified and sent back to the parent process. Although a rather simple program, it demonstrates synchronization and the implementation of pipes.

Project 3 is a more in depth look at Java threads. Besides giving students experience using threads, this project directly contrasts Project 2's use of processes. Threads share data in a very different way than processes. Project 3 gives students a basic understanding of threads by designing a Java program using threads that adds and multiplies all the numbers 1 through N. In addition to this simple threading program, the student is asked to then write a more complex program multiplying two matrices. A separate thread is used to multiply each element of the solution matrix. The students are then challenged to rewrite this program in C by using *PThread*. A more challenging multithreading project — modeling of one of the OS classic examples, Producer-Consumer problem — was given as a bonus project.

CONCLUSIONS AND FUTURE WORK

Students showed great interest in these projects, particularly those involving practice with virtual machines and Linux -- updating the Linux kernel and the addition and testing of new system calls. In addition to everyone completing the first two projects, many students provided positive feedback. In order to better design for future projects, we gave another survey to the students after they completed the above projects. We invited students to critique their learning experiences and we asked for suggestions to help with future modifications of the projects. The following gives a brief summary of the survey.

- Virtualization and virtual machines: more than 80% of students replied that the projects *greatly* helped them understand the related subjects and use the latest technology. About 85% of students used VMware for the projects, while others used different virtualization software packages with success.
- Linux and open source software: more than 80% of students considered that the projects provided a good opportunity for them to learn Linux and to make use of open source software. More than half of them said that without the projects they would not have even considered using Linux.
- C programming language: half of the students acknowledged that the projects increased their knowledge of the C programming language.
- OS kernel and system call: more than 85% of students responded that the projects helped them not only understand the OS internal structures "in a very elementary way" but also learn the basics of system programming. Many of them recognized that without the projects they could not fully understand the kernel functions of the operating system.
- Additional comments: we also asked students to provide feedback on other aspects of the projects. Many of them recognized the "complexity" of the operating system (it takes about "3-4 hours to compile" — quoted from a survey) and the superior functionality of the Linux command-line to the GUI. They particularly mentioned the thread/process-related projects, which greatly reinforced their understanding of the concepts of process and threading.

Students not only favorably responded to the survey questions but also provided many suggestions that we will benefit from in future versions of these projects. We will continue to use the projects but will spend a bit more time teaching the basics of Linux before assigning the Linux related projects. We will first ask students to do some investigation on the topic of virtualization and adopt different virtualization solutions. We will design more projects in other areas of OS such as multithreading program but continue to have each project build off the knowledge gained from previous projects.

REFERENCES

[1] ACM and IEEE Computer Society, *Computer Science Curriculum 2008: An Interim Revision of CS 2001,* http://www.acm.org//education/curricula/ComputerScience2008.pdf, 2008.

[2] Bergman, M., Low-cost compute clusters in virtualized lab environments, *Journal Computing Sciences in Colleges*, 25 (1), 159-166, 2009.

[3] Bower, T., Experiences with virtualization technology in education, *Journal Computing Sciences in Colleges*, 25 (5), 311-318, 2010.

[4 Brylow, D., An experimental laboratory environment for teaching embedded operating system, *Proceedings of the 39th SIGCSE technical symposium on Computer Science education*, 192-196, 2008.

[5] Eastman, E. G. Exploring Linux as an operating system in the CS curriculum, *Journal Computing Sciences in Colleges*, 21 (4), 83-89, 2006.

[6] Gaspar, A.,Langevin, S., Armintage, W. D., Rideout, M., Enabling new pedagogies in operating systems and networking courses with state of the art Open Source kernel and virtualization technologies, *Journal Computing Sciences in Colleges*, 23 (5), 189-198, 2008.

[7] Hickman, G. D., An overview of virtual machine (VM) technology and its implementation in I.T. student labs at Utah Valley State College, *Journal Computing Sciences in Colleges*, 23 (6), 203-212, 2008.

[8] Laadan, O., Nieh, J., Operating system virtualization: practice and experience,, *Proceedings of the 3rd Annual Haifa Experimental Systems Conference (SYSTOR*, 10), 2010.

[9] Li, P., Selecting and using virtualization solutions--our experiences with VMWare and VirtualBox, *Journal Computing Sciences in Colleges*, 25 (3), 11-17, 2010.

[10] Shade, E., Operating Systems on a Stick, *Journal Computing Sciences in Colleges*, 24 (5), 151-158, 2009.

[11] Silberschatz, A., Galvin P. B., Gagne G., *Operating System Concepts*, 8th ed. John Wiley & Sons. Inc, 2009.

[12] Stackpole, B., The evolution of a virtualized baboratory environment, *Proceedings of the 2008 ACM Conference on Information Technology Education (SIGITE¡|08)*, 243-247, 2008.

ACKNOWLEDGEMENT

The authors of this paper would like to thank Professor Joseph Kasprzyk, chairperson of the Computer Science Department of Salem State University, for his support and guidance in the writing of the paper. The corresponding author of the paper is Beifang Yi (byi@salemstate.edu).