One-Click Distribution of Preconfigured Linux Runtime State

Richard Potter
Japan Science and Technology Corporation
potterr@acm.org

Abstract

Checkpointing virtual machines shows potential for allowing a user to download, install, and initialize a complete software environment by selecting a web page link. Starting with the open source User-Mode Linux project, we added checkpointing functionality and integrated the standard xdelta and curl utilities to handle compression, expansion, and download. The current system allows researchers and others to configure Linux-based software demonstrations in detail and save them as snapshots that can be as small as a few megabytes. Users can then easily replicate every detail of the demonstrations without worrying about compilation, installation, dependencies or initial setup.

1. Introduction

Downloading and trying out a software demonstration can be frustrating, because users must often search for and install supporting programs and compatible libraries. Additionally, users must initialize and setup the unfamiliar software such that it demonstrates something interesting.

The creator of the software demonstration, for example a researcher, can reduce some of the frustrations by reducing dependencies, providing automatic installation scripts, and making the user interface simple enough for a novice user. Clear instructions that describe how to install and use the demonstration can help. However, it is burdensome for the researcher to account for all the environments of potential users or all the possible ways that users might misinterpret the setup instructions.

Virtual machines can solve these problems, because they make it possible for users to have the exact same virtual environment, even if the users' actual host hardware is different. The researcher only needs to install and initialize the demonstration in a virtual machine and then create a checkpoint. Library dependencies only need to be resolved for one virtual machine, and it can be easier for the researcher to setup the demonstration than to explain how to do it. Users can then download and restore the checkpointed machine and experience the demonstration with minimal distraction.

We have been exploring how to create and streamline such a solution from open source components so that it is possible for users to restore complete preconfigured Linux environments by simply clicking on a web page link. The central design problems were selecting the virtual environment, performing the checkpointing, compressing the checkpoints, and downloading and restoring the checkpoints from a web browser.

2. SBUML

The first three problems were solved from our previous work on SBUML [2,4], the computation Scrapbook for User-Mode Linux (UML). UML [5] is a port of the Linux kernel to Linux system calls and resources. For example, whereas a hardware-based kernel might use a SCSI device for block storage, UML uses files in the host Linux. Similarly, UML simulates RAM by memory mapping files on the host. In fact, most of UML's state ends up in files on the host. To make SBUML, we extended UML so that it places all these files in one directory and also keeps track of the small amount of remaining state such as open file descriptors, process registers, and signal state. To checkpoint the system, SBUML simply tells all UML processes to exit any system calls that might modify the file descriptors and then makes a copy of the directory full of state.

Because each snapshot contains all file systems, processes, kernel and user state, the raw state can be several hundred megabytes. Fortunately, UML has a copy-on-write (COW) block driver, so that it is only necessary to save file system *changes* in each snapshot. Therefore, raw snapshot sizes start at the size of the RAM used in the UML machine. If there are few disk changes, snapshots can be as small as 32M or 64M.

To reduce this number more, SBUML is integrated with the xdelta [7] utility so that only the overall binary changes between snapshots need to be saved. This makes it possible to create interesting snapshots that are

only a few megabytes in size, and in some cases as small as 100,000bytes. These sizes can be practical to download even with slow modem connections.

3. Download and Restore

When restoring a snapshot of a given name, SBUML looks for a matching snapshot in a designated default snapshot directory. When given a list of external URLs, SBUML will also search though the external locations whenever the snapshot is not found in the local default snapshot directory. The curl utility is then used to download the snapshot directory to inside the default snapshot directory, where it serves as a cached copy.

If the snapshot has been delta compressed, information about which snapshot it is delta compressed against (i.e. the parent snapshot) is determined and used to automatically do the expansion. If the parent snapshot is not in the default snapshot directory, it is searched for in the external URL locations and automatically downloaded and recursively expanded if it too has been delta compressed.

The web page link is created by linking to a small text file that includes the snapshot name and the external URLs where it can be found. When the user clicks on the link, this information is passed to a simple web browser helper application, which then passes it on to SBUML's restore command. If VNC server is running inside the restored snapshot, VNC viewer is automatically started and attached so that snapshots can have graphical user interfaces.

4. Performance

As a concrete example, consider a snapshot designed to demonstrate how the `socket` system call is used. The snapshot has the source code for the `nc` (netcat) utility compiled with debugging symbols. Two copies of emacs are running, each with a `gdb` debugger session for netcat. 15 breakpoints have been setup in each debugger so that the program will stop on every socket related system call, such as `bind`, `accept`, etc. Windows are positioned so that both debuggers can be viewed at the same time. Furthermore, the command line necessary to start each netcat instance is pre-typed along with necessary parameters so that all the user has to do is press return in each debugger to have one instance establish a socket connection with the other.

The raw size for this snapshot is about 43MB (32MB for RAM, 11MB for disk changes from installing and compiling netcat). When delta compressed against a snapshot of a freshly booted 32MB machine, the snapshot size reduces to 8.4MB. After downloading (42 seconds on a fast ADSL line), it takes 7 seconds to decompress and 11 seconds to restore on a 600Mhz Thinkpad X20 with 320MB of RAM.

5. Ongoing Work

The previous example could be used to annotate the man page for the `socket` system call. Manual pages for programming language components and even end-user application features in spreadsheets and word processors could be annotated similarly. Since all the snapshots for documenting a single application would be similar, delta compression could be very effective. However, this all assumes that the user already has the backing hard disk images and parent snapshots stored locally. For the above example, this amounts to about 700MB of space to hold a RedHat 7.2 installation. The Hash Copy techniques demonstrated in [3] is one possible solution direction that could reduce the need for the initial large download. Use of the Self-certifying File System as in [6] is another interesting possibility.

One issue users do have to worry about before clicking on the snapshot link is security. Although the downloaded code runs entirely in user-mode, stronger security is desirable, such as running the UMLs in a chroot environment. In addition, high-level security specifications, such as the Secure Software Circulation Model [1], could keep the system flexible while providing the users with clearly defined security tradeoffs.

6. References

[1] K. Kato and Y. Oyama, "SoftwarePot: An Encapsulated Transferable File System for Secure Software Circulation," *Proc. of the International Symposium on Software Security 2002*.

[2] O. Sato, R. Potter, M. Yamamoto and M. Hagiya. "UML Scrapbook and Realization of Snapshot Programming Environment," *Proc. of the International Symposium on Software Security 2003*.

[3] C. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. S. Lam, and M. Rosenblum, "Optimizing the Migration of Virtual Computers," *Proc. of the 5th Symposium on Operating Systems Design and Implementation (OSDI), 2002*.

[4] http://sbuml.sourceforge.net/

[5] http://user-mode-linux.sourceforge.net/

[6] UML for Knoppix: http://unit.aist.go.jp/it/knoppix/uml/index-en.html

[7] http://sourceforge.net/projects/xdelta/