A Framework for the Performance Evaluation of Operating System Emulators

by

Joshua H. Shaffer

A Proposal Submitted to the Honors Council

For Honors in Computer Science

15 October 2003

Approved By: _____

Luiz Felipe Perrone
Thesis Adviser

_____

Garry Haggard
Chair, Department of Computer Science

1 Introduction

Most modern operating systems are capable of seamlessly coexisting in heterogeneous computing environments. General industry trends have led to the standardization of many of the essential components of an operating system, such as network protocols and file formats. However, application binaries remain incompatible across operating systems and hardware platforms. Application developers must decide which platforms they wish to target, and in doing so will prevent users of other systems from running their software. As a result, specialized applications may not be available on a user's platform of choice, and he or she will often either abandon the preferred platform or find a way to work without the applications in question. An alternative solution is to have an application on the user's system emulate the second platform, thus allowing the necessary programs to be run.

There are two main classes of emulators. The first emulates every aspect of the target system at the hardware level, providing an environment in which an instance of the required operating system can be run as a process on the host machine. The second takes a higher level approach, intercepting calls destined for the target operating system and recreating their intended effect on the host system. The latter class of emulators may be combined with processor emulation to provide a virtual machine in which an application can be run on an architecture other than the one for which it was originally compiled. Each of these types of emulators has its own set of benefits, but no standard method exists for comparing different implementations. It is this deficiency that I will address through the creation of a framework for the performance evaluation of operating

system emulators.

2 Background

The traditional solution to emulation requires that every major component of the target computer's hardware be implemented in software. This approach allows a copy of the target operating system to be run within the emulator, and thus for applications designed for that system to be run in their native environment. Because a full copy of the target operating system is run on the emulator, a high level of compatibility can generally be achieved. However, this compatibility comes at the expense of speed, which is limited by the requirement that many components which already exist on the host machine be emulated rather than used directly. For example, memory systems, hard disks, and graphics controllers are re-implemented in software, adding significant overhead to the already complex task of emulating the target processor [3].

The higher level approach to emulation removes the necessity for such redundant systems by mapping operating system API calls directly to frameworks that exist on the host system. A virtual machine of this type still requires the emulation of the target processor when it differs from the host's processor, but other systems need not be emulated. This approach may be more complex than full system emulation, as it requires a detailed knowledge of the target API, but it has the potential to provide a more efficient method for running the target system's binaries. Benefits to this approach over full system emulation include the ability to provide the emulated software with the host

operating system's native look-and-feel, while removing the need to install a copy of the target operating system.

3 Project Description

This project will focus on the creation of a framework for the evaluation of an emulator's implementation. Performing this evaluation for various types of emulators would provide the basis for a comparison of their implementations. Although the framework will allow for the examination of emulators on many platforms, experimental results will only be provided for emulators that run Microsoft Windows applications on Mac OS X.

Microsoft's Virtual PC is by far the most popular PC emulator available on the Macintosh platform. It is a reasonably fast full system emulator, and its integration with other Macintosh applications is unmatched. Bochs, an open source alternative to Virtual PC, provides functionality similar to its commercial cousin and is available for free. However, it offers lower compatibility and lacks any significant integration with Mac OS X. For these reasons, only Virtual PC will be tested in this study, although it should be noted that such alternatives do exist.

SPIM [2], an emulation of the MIPS 2000 instruction set, and Wine [4], a custom implementation of the Windows API, are examples of the two components of a virtual machine. However, no application exists which combines processor emulation with API re-implementation to allow Windows applications to be run on a Macintosh. I have created, on my own time, a partial implementation of a virtual machine, code-named

Champagne, capable of executing 32-bit Windows binary applications. This system will serve as the second emulator used in the demonstration of the framework to be developed for performance comparison.

4 Methodology

The framework to be developed will stress a wide array of components which are critical to emulator performance. The tests will include groups targeting specific aspects of the emulated environments such as the graphics, memory, disk, and networking subsystems [1]. Although it is important to perform comparisons in these individual areas to determine where specific optimizations may be necessary, an analysis of the performance that can be expected when running real applications will be of greater benefit to end users. Due to their reliance on a wide range of components within each emulator, algorithms such as audio or video compression will be used to provide a broad view of real-world performance.

Extra emphasis will be placed on identifying and evaluating areas in which performance varies greatly between the major types of emulators. For example, memory management, hard disk access, and network operations should be much faster in a virtual machine than in a full system emulator. In general, components that in full emulators have both emulated and native layers will likely see the largest increase in performance. Memory management, in particular, should be more efficient, because rather than emulating the target architecture's physical memory as a layer on top of the host's native

system, virtual machines can provide memory by calling native allocation routines directly. Similar arguments can also be applied to many other components, with the expected net result being a significant improvement in runtime speed. To obtain accurate results, performance and runtime speed will be defined as a test's total execution time, thus including all of the overhead introduced by the emulator [2].

5 Conclusion

The main goal of this project is to create a framework which can be used to compare the performance of different emulators, both in specific areas and in general usage cases. Such a framework would allow emulator developers to benchmark target areas of their applications in order to determine where optimizations are most needed. It would also provide end users with an idea of the performance to expect from different emulator implementations when running various types of software.

The project will draw on knowledge gained in almost every aspect of the computer science curriculum. An understanding of microprocessor architecture is necessary for the analysis of CPU emulation speed and efficiency, with an emphasis on branch prediction and instruction caching. Algorithm design and analysis will play a large role in the creation of the suite of tests and the analysis of the results. Finally, an understanding of operating systems and their major components is necessary for every aspect of the project. With such a wide variety of topics being covered, the development and testing of the framework will be a culmination of my experiences at Bucknell.

References

[1] J. Bradley Chen, et al. "The Measured Performance of Personal Computer Operating Systems." *The Proceedings of the 15th ACM Symposium on Operating System Principles.*

[2] Patterson, David A. and Hennessy, John L. *Computer Organization & Design: The Hardware/Software Interface.* San Francisco: Morgan Kaufmann Publishers, Inc., 1998.

[3] Traut, Eric. "Building the Virtual PC: A software emulator shows that the PowerPC can emulate another computer, down to its very hardware." Byte. November 1997. Accessed 8 October 2003. [`http://www.byte.com/art/9711/sec4/art4.htm`].

[4] Wine Developer's Guide. Section II: Wine Architecture. [`http://www.winehq.com/docs/winedoc-pdf.tgz`]