

# The Use of Microcode Instrumentation for Development, Debugging and Tuning of Operating System Kernels

Stephen W. Melvin  
Yale N. Patt

Computer Science Division  
University of California, Berkeley  
Berkeley, CA 94720

## ABSTRACT

We have developed a tool based on microcode modifications to a VAX 8600 which allows a wide variety of operating system measurements to be taken with minimal perturbation and without the need to modify any operating system software. A trace of interrupts, exceptions, system calls and context switches is generated as a side-effect to normal execution. In this paper we describe the tool we have developed and present some results we have gathered under both UNIX 4.3 BSD and VAX/VMS V4.5. We compare the process fork behavior of two different command shells under UNIX, look at context switch rates for interactive and batch workloads and generate a histogram for network interrupt service time.

## 1. Introduction

Two basic approaches to performance analysis are modeling and data collection. While modeling has important applications, it can become intractable for complex systems unless overly simplistic assumptions are made. The interactions that occur on a large multiprogrammed computer with a modern operating system are so varied and complex that empirical measurements are essential. Furthermore, it is important to carefully consider the quality of the data being collected and the cost of collecting it.

There are three fundamental approaches to data collection: hardware, software and microcode. Hardware monitors can take measurements with virtually no effect on the system. However, they can be inflexible, limited in resources, expensive and cumbersome to operate. Alternatively, the operating system kernel can be

modified to collect data which, although satisfactory for some types of measurements, can be impractical and/or cause an unacceptable perturbation of the measurement for others. In addition, modifying the kernels of most operating systems is not a matter to be taken lightly. It is often difficult to determine the effect that a seemingly small change can have on other parts of the kernel.

Most of the advantages of both hardware and software methods can be achieved with a microprogrammed measurement gathering technique. By modifying the microcode, measurements can be taken with a very small effect on the system. In addition, microcode-based systems can be flexible and easy to use. Once the core microcode is installed that implements the data collection tool, everything else can be under software control. Furthermore, since the data collection takes place below the operating system there no need to modify the kernel and the same measurement can be taken on different operating systems.

The idea of using microcode to gather measurements has been around for a long time. The earliest mention of which we are aware was by Halbach in 1971 [5]. In this two page note, Halbach discusses the gathering of trace information through the use of microcode modifications. Armbruster discusses the gathering of instruction traces in [2] and Grätsh and Kästner provide a brief history of firmware monitoring in [4]. Chroust, Kreuzer and Stadler discuss a microprogrammed page-fault monitor in [3] and Agarwal, Sites and Horowitz discuss a microcode-assisted address tracer in [1].

We have implemented a microcode based event tracer which we call SPAM (for System Performance Analysis using Microcode) which differs from these previous methods in two important ways. First, it is specifically geared toward gathering real-time sensitive data. The quantity and type of information being gathered is carefully chosen to minimize perturbation of the system. Second, SPAM is a general facility that allows a wide variety of data to be collected rather than a specific tool to collect one particular type of data. We have also implemented an interface to the microcode to allow SPAM to be used without the need for microcode expertise or specific hardware knowledge.

SPAM is based on microcode modifications to a VAX 8600 which include additional machine level in-

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 ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

