L4oprof: A System-Wide Profiler Using Hardware PMU in L4 Environment

Jugwan Eom, Dohun Kim, and Chanik Park

Department of Computer Science and Engineering
Pohang University of Science and Technology
Pohang, Gyungbuk 790-784, Republic of Korea
{zugwan,hunkim,cipark}@postech.ac.kr

Abstract. The recent advance of L4 microkernel technology enables building a
secure embedded system with comparable performance to a traditional
monolithic kernel-based system. According to the different system software
architecture, the execution behavior of an application in microkernel
environment differs greatly from that in traditional monolithic environment.
Therefore, we need a performance profiler to improve performance of the
application in microkernel environment. Currently, L4’s profiling tools
provide only program-level information such as the number of function calls,
IPC, context switches, etc. In this paper, we present L4oprof, a system-wide
statistical profiler in L4 microkernel environment. L4oprof leverages the
hardware performance counters of PMU on a CPU to enable profiling of a wide
variety of hardware events such as clock cycles and cache and TLB misses. Our
evaluation shows that L4oprof incurs 0~3% higher overhead than Linux
OProfile. Moreover, the main cause of performance loss in L4Linux
applications is shown compared with Linux applications.

Keywords: L4 microkernel, performance analysis, performance measures,
performance monitoring, statistical profiling, hardware PMU.

1 Introduction

To analyze a program’s performance, its execution behavior is investigated by
monitoring runtime information. Monitoring program execution is important, because
it can find the bottlenecks and determine which parts of a program should be
optimized. The type of collected information depends on the level at which it is
collected. It is divided into the following two levels.

1. The program level: the program is instrumented by adding calls to routines
   which gather desired information such as the number of time a function is called, the
   number of time a basic block is entered, a call graph, and an internal program state
   like queue length changes in the kernel block layer.

2. The hardware level: the program does not need to be modified. CPU architecture
   like caches, pipelines, superscalar, out-of-order execution, branch prediction, and
   speculative execution can lead to large differences between the best-case and the

© Springer-Verlag Berlin Heidelberg 2007
worst-case performances, which must be considered to increase program performance. Most of current CPUs have a hardware component called PMU (Performance Monitoring Unit) for programmers to exploit the information on CPU. The PMU measures the micro architectural behavior of the program such as the number of clock cycles, how many cache stalls, how many TLB misses, which is stored in performance counters.

The goal of performance analysis is to find where time is spent and why it is spent there. Program-level monitoring can detect performance bottlenecks, but finding their cause is best solved with hardware-level monitoring, which may result from function calls, algorithmic problems, or CPU stalls. Therefore, the two levels of monitoring must be used complementarily for proper performance analysis.

The recent advance of L4 microkernel technology enables building a secure embedded system with comparable performance to a traditional monolithic kernel-based system. Industry such as QUALCOMM sees L4 microkernel’s potential as a solution to the security problems of embedded systems. In a microkernel-based system, the basic kernel functions such as communication, scheduling, and memory mapping are provided by the microkernel and most of OS services are implemented as multiple user level servers on top of the microkernel, in which the execution behavior of an L4 application differs greatly from that in traditional monolithic environment. Therefore, we need a performance profiler to improve performance of the application in microkernel environment. However, L4 microkernel provides only program-level profiling tools which do not utilize the PMU information that is also valuable for fine-grained performance profiling, which enables to locate the cause of performance inefficiency in an application.

In this paper, we present L4oprof, a system-wide statistical profiler in L4 microkernel environment. L4oprof leverages the hardware performance counters of PMU on a CPU to enable profiling of a wide variety of hardware events such as clock cycles and cache and TLB misses without program modification. L4oprof can profile applications in the system and the L4 microkernel itself. This paper also shows the main cause of performance loss in L4Linux applications compared that in Linux applications. L4oprof has been modeled after the OProfile [5] profiling tool available on Linux systems.

The remainder of the paper is organized as follows. Section 2 describes related work. We describe the aspects of L4 and OProfile as background for our work in Section 3. Section 4 describes the design and implementation of L4oprof. Then, we present L4oprof’s performance in Section 5. Finally, we summarize the paper and discuss the future work in Section 6.

2 Related Work

Several hardware monitoring interfaces [8, 9] and tools [6, 7, 10, 15] have been defined for using hardware performance monitoring on different architectures and platforms. Xenoprof [15] is a profiling toolkit for the Xen virtual machine environment, which has inspired the current L4oprof’s approach.

In L4 microkernel-based environment, a few performance monitoring tools are available. Fiasco Trace Buffer [11] collects the kernel internal events such as context