Static worst-case execution time analysis of the μC/OS-II real-time kernel

Mingsong LV (✉)1, Nan GUAN1, Qingxu DENG1, Ge YU1, WANG Yi1,2

1 Institute of Computer Software, Northeastern University, Shenyang 110819, China
2 Department of Information Technology, Uppsala University, Uppsala S-75105, Sweden

© Higher Education Press and Springer-Verlag Berlin Heidelberg 2010

Abstract  Worst-case execution time (WCET) analysis is one of the major tasks in timing validation of hard real-time systems. In complex systems with real-time operating systems (RTOS), the timing properties of the system are decided by both the applications and RTOS. Traditionally, WCET analysis mainly deals with application programs, while it is crucial to know whether RTOS also behaves in a timely predictable manner. In this paper, static analysis techniques are used to predict the WCET of the system calls and the Disable Interrupt regions of the μC/OS-II real-time kernel, which presents a quantitative evaluation of the real-time performance of μC/OS-II. The precision of applying existing WCET analysis techniques on RTOS is evaluated, and the practical difficulties in using static methods in timing analysis of RTOS are also discussed.

Keywords  worst-case execution time (WCET), real-time operating systems (RTOS), μC/OS-II, static analysis

1 Introduction

Hard real-time systems are those systems in which tasks must meet their deadlines; otherwise, there will be disastrous consequences. Timing correctness of hard real-time systems is traditionally guaranteed by a hierarchical off-line analysis framework. First, worst-case execution time (WCET) analysis is used to obtain the execution time of tasks in the worst case, and then schedulability analysis uses these results to decide whether all the tasks are schedulable. There are two important properties to describe the usefulness of an analysis technique: safety and accuracy. The results are safe if no actual execution of the program exceeds the estimated time. Also, the estimation is said to be more accurate if it is closer to the real maximal execution time of the program. Soft real-time systems do not always have safety requirements, but hard real-time systems allow no underestimation. The accuracy of the estimation is also critical, since too pessimistic estimations lead to over-design and low task acceptance ratio. Using only toy benchmarks is inadequate to justify the usefulness of the analysis techniques, so it is important to test the techniques on real-life programs.

Traditionally, WCET analysis is mainly applied to application programs and has achieved success in industry (e.g. aiT1). While complex real-time systems are composed of both applications and real-time operating systems (RTOS), and the timing properties of the system are decided by both parts. In order to obtain WCET estimations for a whole system, timing analysis should be performed not only on application programs, but also on RTOS services. Although it does not make much difference between application code and RTOS code when they are compiled to binaries, (the input to static WCET analysis), analyzing RTOS is generally harder than analyzing applications with no library/system calls, since the behaviors of RTOSes are much more complex. Simply applying static analysis techniques designed for applications may yield low-quality or even incorrect results.
In an RTOS, certain parts of the system code are particularly time-critical. An example is the DI regions which execute with the interrupts turned off. The execution of DI regions directly affects the response of the system to external events. So it is useful to quantitatively bound the execution time of DI regions, which will further help RTOS designers to identify the bottlenecks and then improve the responsiveness of the RTOS.

This paper is an extension of our previous conference paper [1], in which Chronos, a static WCET analysis tool, is used to obtain the WCETs of the system calls (or APIs) of the μC/OS-II real-time kernel. The main enhancement of this paper is the WCET analysis of the DI regions of μC/OS-II (also using static analysis techniques). The purpose of this research is not only to test the accuracy of the estimation, but also to investigate the practical difficulties in applying static analysis techniques to real-life RTOS code. In our experiments, we successfully checked the WCET of 61 (out of 79) μC/OS-II system calls and all the DI regions residing in these system calls. Compared to simulation-based timing analysis methods, the quantitative results obtained by WCET analysis can give a safer picture of the real-time performance of an RTOS. In our practice, we find that some traditional WCET analysis techniques, such as those adopted in Chronos, are far from adequate in characterizing the timing properties of an RTOS. The problems found from our research include a lack of parametric timing analysis techniques in RTOS analysis and incorrect results in the presence of context switches, etc.

The rest of the paper is organized as follows. Section 2 gives an introduction to μC/OS-II. The experimental settings applied in our research are introduced in Section 3. Section 4 elaborates how the WCETs of the system calls and the DI regions are obtained. Analysis results and some further issues are discussed in Sections 5 and 6, respectively. Section 7 lists related research, and the paper is concluded in Section 8.

2 The μC/OS-II real-time kernel

μC/OS-II[2] is an open-source real-time kernel designed by Micrium, Inc. The μC/OS-II kernel is designed to be efficient with a small footprint. Although it does not have as many features as other RTOSes such as RTEMS and VxWorks, it has nearly all the standard RTOS capabilities: (1) Priority-based preemptive scheduling; (2) Inter-task communications via semaphore, mutex, message queue, and message box; (3) Time management; (4) Simple memory management.

μC/OS-II is one of the most widely used real-time kernels in industry: it has been licensed by hundreds of real-time embedded systems companies, and the products span multiple domains, including network management devices, handheld devices, and embedded monitor and control systems. μC/OS-II is also certified in avionics products by FAA for use in commercial aircrafts. We believe it is of great practical use to evaluate the realtime performance of μC/OS-II quantitatively due to its prevalence in industry.

3 Experimental settings

This section gives the detailed experimental settings applied in our experiments: the configuration of μC/OS-II, and the WCET analysis tool adopted.

3.1 μC/OS-II configuration

The μC/OS-II real-time kernel has a small footprint with 11 C files and one header file, which contain 79 system calls spanning 9771 lines of code. μC/OS-II allows developers to customize these features according to their requirements. In this paper, we exclude some features that are not the core functions of the system. Excluded features are: name management, the statistic task, task profiling, and debugging. Some trivial functions, such as the dummy function, are also excluded. We refer readers to our technical report[2] for further details.

In order for μC/OS-II to run on a specific processor, one must first port the system to the target instruction set. μC/OS-II requires that functions to do context switch and disable/enable interrupts should be rewritten for different architectures. We find that these architecture-specific functions are all single-path functions, and they do not incur extra difficulty in the analysis. Additionally, Simplescalar the adopted simulator in the experiments, does not fully support full-functional simulation of operating systems. So without loss of generality, we replace these functions with dummy functions in our experiments.

1) http://www.micrium.com, 2009