Impact of RTOS Parameters on End-to-End Timing Performance

Antino Kim and Kang G. Shin
Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122
email: {kheehyon,kgshin}@eecs.umich.edu

Abstract

For accurate end-to-end (e2e) analysis of a real-time embedded system (RTES), we need a method to precisely reflect the effects of the underlying real-time operating system (RTOS). In timing analysis, RTOS can be viewed as a matrix of parameters that could affect the timing of the RTES. In this paper, we consider various RTOS parameters that may affect the temporal behavior of the RTES, and propose RTOS modeling as a viable approach to account for the impact of RTOS on e2e timing.

1. Introduction

To better utilize the underlying computing resources, real-time operating systems (RTOSs) are widely used in the real-time embedded system (RTES) field. In an early design phase of RTES development, the system designers perform rapid prototyping to test the validity of the system. It is the role of end-to-end (e2e) timing analysis to aid the system designers in making a decision on which available RTOS to use. Therefore, it is important to correctly reflect the temporal effects of RTOS on the overall system. However, RTOS itself is highly complicated, and it is not obvious how to account for its impact on e2e timing.

In this paper, we propose RTOS modeling based on profiling to account for the impact of RTOS on e2e timing. We view RTOS as a matrix of parameters that could affect the timing of the RTES. We first discuss RTOS parameters and how they may affect the temporal behavior of the RTES, and then describe our approach in modeling the RTOS along with steps we plan to take.

2. Problem

There are many parameters of RTOS that have great impact on e2e timing, including timer granularity, choice of scheduler, IPC facilities, and so on. Table 1 lists some of the major RTOS parameters. We have tried to maintain the description of RTOS as general as possible while bringing out some of the important characteristics from the crowd. We will discuss some of these parameters in more detail.

Table 1. List of example RTOS parameters that affect the timing of RTES

• Scheduler: Scheduling policy, associated timing overhead, support for task preemption, scheduler triggering events (e.g., blocking of a task, timer ISR, etc.) • Priority inversion resolution protocols: Priority inheritance protocol, priority ceiling protocol, disabling preemption during access to shared data [4]
• Timer granularity
• Preemptability & preemption points of kernel services • Kernel thread priority
• Set of system calls and their duration • Task synchronization primitives (e.g., lock, semaphore, mutex, etc.) • Set of IPC facilities (message passing, barrier, etc) and their characteristics (i.e., blocking / non-blocking)
• Set of ISRs (Timer ISR, I/O ISR) and their duration • Effect of splitting I/O interrupt into *top/bottom half*

Scheduler is a mechanism that determines which task gets to run at a certain instant, and it is usually done by using some notion of priority. Priority inversion resolving protocols [3] may cause unintended change in task priority, and therefore, affect scheduler's decision. Moreover, according to [3], priority inheritance protocol and priority ceiling protocol have different timing overheads. Also, the choice of scheduling policy brings different overheads to the system. For example, it is commonly known that Earliest-Deadline-First (EDF) scheduling has higher run-time overhead than Rate-Monotonic (RM) scheduling, while RM induces a higher schedulability overhead.

Timer granularity determines how finely the time can be managed. Together with the choice of a scheduler, timer granularity may significantly affect the number of context switches, thus the accumulated

overhead. In addition, the timer granularity would determine the accumulated overhead of timer interrupt handler.

Other parameters listed in the table are all important factors to resolve in order to conduct a precise e2e analysis. Especially, effects of I/O interrupt handlers would be an interesting issue to study thoroughly. For a better response time, I/O interrupts are usually divided into *top-half* and *bottom-half* handlers [1]. While it may be a reasonable approach in general purpose OSs, arbitrary delay of bottom half processing introduces another source of timing uncertainty.

While there are many factors in RTOS that affect e2e timing of a system, it is unclear how to reflect the effects of those parameters when designing a system. It would be wasteful to purchase a RTOS and port applications on it just to see whether the given implementation satisfies e2e timing requirement. It would be more useful for the system designers to have a way to perform rapid prototyping by adjusting the knobs on numerous RTOS parameters at design time.

3. Proposal

We propose RTOS modeling based on its parameters as a viable approach to studying the effects of RTOS on e2e delay. Basically, we abstract the impact of RTOS on e2e timing as a matrix of parameters that we mentioned above. While there have been numerous efforts in modeling RTOS [2, 5], there has not been a solution that is comprehensive enough for practical use. In [2], the authors model RTOS by categorizing RTOS parameters into *behavior model* and *timing model*, where the behavior model is characterized by scheduling policy and preemptability, and the timing model is represented by scheduling duration and context switch overhead. These are important parameters of RTOS, but we need a more comprehensive list to make the model fully capture the temporal effects of RTOS. In [5], the authors present a configurable RTOS modeling tool, and our proposal is, in principle, similar to their work. However, while [5] deals with many of the major RTOS parameters, it ignores essential factors of scheduling such as priority inversion resolution protocols. The model provides a generic priority driven scheduler without the policy of how the priority of a task is resolved in the scheduler (e.g., RM, EDF). Moreover, only one application is run in the experiment, so there is no conclusive evidence that their model fully captures the impact of RTOS in cases where multiple processes run concurrently.

Borrowing the concept of RTOS modeling, we plan to develop a refined model by conducting a full study on the parameters that could affect the temporal behavior of the RTES. The model should be flexible enough to accommodate new ideas in the future. The authors of [5] suggest a list of requirements that RTOS modeling should fulfill. We will investigate on how to represent the parameters of RTOS. For the scheduler parameter, we will develop a way to describe its policy in a systematic yet extensible way. For parameters such as timer granularity or system call duration, we may leave them open for users to specify. This will give users more flexibility. When modeling an existing RTOS, we may use some of the available measurements or make additional measurements.

With a constructed model, we will perform a sensitivity test to see the degree of effect that each parameter has on the RTES. Sensitivity analysis may reveal some of the unforeseen effects of RTOS parameters. Especially, variations in combination of parameters may produce synergy effects that would be hard to predict by analyzing each parameters separately. RTOS modeling will facilitate such multi-factorial studies. Conversely, sensitivity test will help us refine the model of RTOS. During the test, we may come to realization that for some applications, a certain set of parameters has minimal effect, and therefore, is irrelevant to the e2e timing. Such conclusion would make the model simpler and easier to analyze.

We believe that a model-based approach in accounting for the impact of RTOS would be effective in rapid prototyping. The model will enable system designers to correctly evaluate the behavior of overall system while considering the impact of RTOS. All these analyses can be done without having to port the tasks onto an actual RTOS. Based on the observations, RTOS model is an effective solution to the problem of accounting for the impact of RTOS on overall timing performance.

4. References

[1] Y. Zhang, and R. West, *Process-Aware Interrupt Scheduling and Accounting*, Proceedings of the 27th IEEE International Real-Time Systems Symposium (RTSS), 2006

[2] R. Moigne, O. Pasquier, and J. Calvez, *A Generic Model for Real-time Systems Simulation with SystemC*, Proceedings of the Design, Automation and Test in Europe Conference and Exhibition Designers' Forum (DATE), 2004

[3] D. Locke, *Priority Inheritance: The Real Story*, http://www.linuxdevices.com/articles/AT5698775833.html,2002

[4] G. Buttazzo, *Hard Real-Time Computing Systems (The International Series in Engineering and Computer Science),* Kluwer Academic Publishers, ISBN 0792399943, 1997

[5] Z. He, A. Mok, and C. Peng, *Timed RTOS Modeling for Embedded System Design*, Proceedings of the 11th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS), 2005