The Journal of Systems and Software 102 (2015) 58–71

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Power-aware scheduling of compositional real-time frameworks

Guy Martin Tchamgoue, Kyong Hoon Kim*, Yong-Kee Jun

Department of Informatics, Gyeongsang National University, 501 Jinju-daero, Jinju 660-701, South Korea

ARTICLE INFO

Article history:
Received 2 July 2014
Revised 10 November 2014
Accepted 13 December 2014
Available online 20 December 2014

Keywords:
Compositional and hierarchical real-time scheduling
Periodic resource model
Periodic task model
Power-aware scheduling

ABSTRACT

The energy consumption problem has become a great challenge in all computing areas from modern handheld devices to large data centers. Dynamic voltage scaling (DVS) is widely used as mean to reduce the energy consumption of computer systems by lowering whenever possible the voltage and operating frequency of processors. Unfortunately, existing compositional real-time scheduling frameworks have been focusing only on efficient scheduling of tasks inside their components given a resource model, providing no interest on power/energy consumption. In this paper, we define the real-time DVS problem for a compositional scheduling framework. Considering the periodic resource model, we propose optimal static DVS schemes at system, component, and task levels. We also introduce component and task level dynamic DVS schemes that take advantage of runtime unused slack times and resource availability to provide even better energy savings. Finally, we provide power-aware schedulability conditions to guarantee the feasibility of each component under DVS for the Earliest Deadline First and the Rate Monotonic scheduling algorithms. Through simulations, we showed that our schemes can reduce the energy consumption of a component by up to 96%.

1. Introduction

Recently, computation and communication have radically shifted to mobile and portable computing platforms with the birth of new devices like smartphones, tablet computers, smart watches, smart robots, wearable computers, unmanned vehicles, and smart autonomous vehicles. However, instead of using low power processors, these devices rather rely on more and more powerful processors to satisfy the needs of their complex and sophisticated software. A common problem encountered by users of these devices is the short battery life. This problem is the direct consequence of the inherent conflict existing in the design goals of these devices (Pillai and Shin, 2001). As mobile systems, they should be designed to maximize battery life, but as smart devices, they need powerful enough processors, which consume more energy than those in simpler devices, thus reducing battery life. Despite the continuous advances in semiconductor and battery technologies that allow microprocessors to provide much greater computation per unit of energy and longer total battery life, the fundamental tradeoff between performance and battery life remains critically important. Thus, at design level, two conflicting problems need to be solved: the complexity of software systems and the power/energy consumption of the required hardware. Nevertheless, the power consumption problem is a challenge in all computing areas including large data centers and cloud computing, where reducing the energy consumption directly impacts the management cost and contributes to a greener computing environment.

To handle the growth and complexity of software systems, the component-based design has been widely adopted as it proposes to decompose a single complex system into independent subsystems or *components* and provides ways to reassemble them into a flexible hierarchically scheduled framework with real-time guarantees to each subsystem. This design facilitates the reuse of components that may therefore be developed in different environments. Compositional real-time scheduling frameworks are generally organized in a tree-like structure and allow components to share resources transparently using different scheduling policies. In the hierarchy, an upper-layer component decides the amount of resource to be supplied to its child components according to a resource supply scheme.

Significant research efforts have so far been made and many compositional scheduling frameworks (Burmyakov et al., 2014; Easwaran et al., 2009; Phan et al., 2010; Shin and Lee, 2008; Tchamgoue et al., 2013) proposed to support various real-time task models. With these frameworks come a variety of resource supply models to govern the resource demand and supply in the system. Unfortunately, these scheduling frameworks have been focusing only on efficient scheduling of tasks inside their components given a resource model, providing no interest on power/energy consumption, which however has become a hot issue in many recent real-time embedded systems.

Thus, number of techniques (Aydin et al., 2004; Niu and Li, 2011; Pillai and Shin, 2001; Seo et al., 2008; Zhuravlev et al., 2013) have

* Corresponding author. Tel.: +82 557721375.
E-mail addresses: guymt@ymail.com (G.M. Tchamgoue), khkim@gnu.ac.kr (K.H. Kim), jun@gnu.ac.kr (Y.-K. Jun).

http://dx.doi.org/10.1016/j.jss.2014.12.031
0164-1212/

been suggested based on the *dynamic scaling voltage* (DVS), which is available on almost all modern microprocessors, to reduce the energy/power consumption of computing systems. However, these techniques cannot blindly be applied to compositional scheduling frameworks, since they all assume the resource to be continuously available, whereas the resource availability in these new systems is dictated by the resource model that may be periodic for example (Shin and Lee, 2003). DVS dynamically adjusts both the supply voltage and operating frequency to reduce the energy consumption of a processor. Thus, one can achieve low performance whenever the full processing speed is not required by simply scaling down the frequency of the processor. Since the power consumption of a processor is a strictly increasing convex function of the supply voltage, the energy dissipated per computation cycle also scales quadratically to the supply voltage (Aydin et al., 2004; Pillai and Shin, 2001), allowing DVS to potentially provide significant energy savings.

In this paper, we address the power-aware scheduling of time-constrained tasks in a compositional real-time scheduling framework. We consider the compositional real-time scheduling framework with periodic resource model (Shin and Lee, 2008), and solve the compositional real-time dynamic voltage scaling (CRT-DVS) problem: *adjust the supply voltage and operating frequency of the processor to minimize the energy consumption of the system while still guaranteeing the deadlines of individual tasks inside each component*. Considering the Earliest Deadline First (EDF) and the Rate Monotonic (RM) scheduling algorithms, we analyze the schedulability of a compositional real-time scheduling framework under DVS. We propose optimal static DVS schemes that assign processor speed at *system*, *component*, or *task* levels to minimize the energy consumption assuming the worst-case workload of each component. Furthermore, we take advantage of unused resources at runtime to introduce dynamic DVS algorithms at *component* and *task* levels for even more energy reduction. Throughout simulations on synthetic workloads, we show that our schemes can reduce the energy consumption of a component by up to 96%. On a real-world avionics benchmark, our dynamic schemes show a significant gain in energy of up to 39.6% on average.

For the remainder, this paper is organized as follows. Section 2 defines the system model including the compositional real-time scheduling framework and the power model, and presents a motivating example followed by the problem definition. Section 3 is about the schedulability analysis of the system. Section 4 focuses on the static schemes, while Section 5 presents the dynamic schemes. Section 6 evaluates the performances of the proposed schemes. Some additional considerations are given in Section 7. The related work is presented in Section 8 and the paper concluded in Section 9.

2. System model and problem definition

This section provides an overview of compositional real-time scheduling frameworks. It also presents the power model and clarifies some assumptions and notations to be used in the article. Finally, a motivation example followed by our problem definition is presented.

2.1. Compositional real-time scheduling framework

In a *compositional* scheduling framework (Shin and Lee, 2008; Tchamgoue et al., 2013), a *component* is the basic scheduling unit and is defined by $C(W, A)$, where W is the workload and A the scheduling algorithm of the component. Components are organized in a tree-like hierarchy where an upper-layer component allocates resources to its child components, as shown in Fig. 1.

In this paper, we assume a compositional scheduling framework for periodic task model (Shin and Lee, 2008). Thus, the workload W of each component $C(W, A)$ consists of n periodic real-time tasks. Each task τ_i is defined by (p_i, e_i), where p_i represents the period of the task and e_i its worst-case execution time. A task τ_i releases a new job every p_i time units. Therefore, the jth job of task τ_i denoted by $J_{i,j}$ should be finished by the next job's release time. The periodic task model is very common in many cyber-physical systems including avionics (Easwaran et al., 2009) and automotive systems (Asberg et al., 2009).

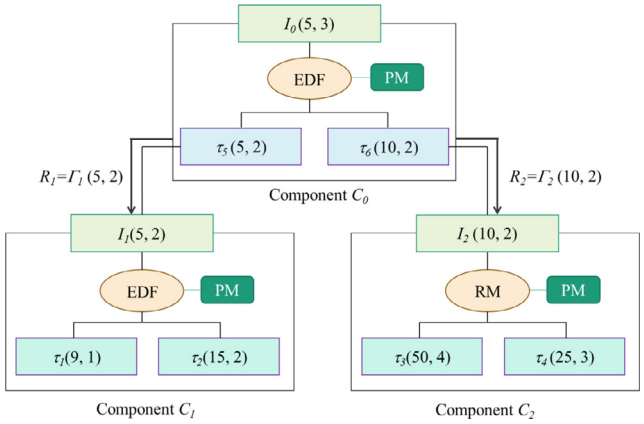

Fig. 1. An example of compositional real-time scheduling framework.

Each component of the framework is abstracted through its *interface* and seeing by its upper-layer component as a single real-time task. This abstraction allows the upper-layer component to schedule its child component without any consideration of their internal real-time requirements. The periodic interface model considered here reduces a component into a new periodic task $I(P, E)$, where P is the period of the task and E its execution time. In Fig. 1 for example, the two tasks of component C_1, scheduled with Earliest Deadline First (EDF), are abstracted into a new periodic task $I_1(5, 2)$. Similarly, component C_2, which is scheduled using the Rate Monotonic (RM) policy, is converted into $I_2(10, 2)$. The upper-layer component C_0 then only focuses on efficiently scheduling these two component tasks, providing them with the appropriate resource supply.

A *resource model* in a compositional real-time scheduling framework specifies the minimum amount of resource to be allocated to a component for it to be schedulable. For this work, we consider the periodic resource model proposed by Shin and Lee (2003). A periodic resource model $\Gamma(\Pi, \Theta)$ guarantees that a resource amount of Θ time units is supplied every Π time units to a component. For example, the resource model $\Gamma_1(5, 2)$ of component C_1 in Fig. 1 guarantees that 2 time units are provided to the component every 5 time units. Similarly, a computational time of 2 units is given to C_2 at every 10 time units.

2.2. Power model

In this paper, we consider a compositional real-time scheduling framework running on a single processor system. We assume the processor to be equipped with dynamic voltage scaling (DVS) mechanism allowing it to continuously adjust its operating frequency f between a minimum frequency level $f_{\min}$ and a maximum frequency level $f_{\max}$ ($f_{\min} \leq f \leq f_{\max}$). We then define the speed level s of a processor as the normalized operating frequency f with respect to the maximum frequency $f_{\max}$, i.e. $s = f/f_{\max}$. Thus, without loss of generality, throughout this paper, we will only refer to the processor speed level s which is well defined between a minimum value $s_{\min}$ and a maximum value $s_{\max}$, $0 \leq s_{\min} \leq s_{\max} = 1$. In this work, the processor speed adjustment occurs at resource release time or during context switching.

Now, let us consider an application with its execution time given by t when the processor is running at its maximum frequency $f_{\max}$, i.e. at maximum speed level $s_{\max} = 1$. Let us assume that the processor is scaled to run at a frequency level f ($0 < f \leq f_{\max}$) which corresponds