Computer Standards & Interfaces xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Computer Standards & Interfaces

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

Energy-aware mixed partitioning scheduling in standby-sparing systems

Zhang Yi-wen[a,b,*]

[a] College of Computer Science and Technology, Huaqiao University, XiaMen 361021, China,
[b] Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang, 110168, China

ARTICLE INFO

Keywords:
Standby-sparing systems
Mixed partition
Energy management
Real time scheduling

ABSTRACT

Previous standby-sparing techniques assume that all tasks don't access to shared resources. In addition, primary tasks and backup tasks are allocated to the primary processor and spare processor respectively. Spare processor schedules tasks with maximum processor speed. Unlike previous techniques, we have studied the problem of minimizing energy consumption and preserving the original reliability for dynamic-priority real-time task set with shared resources in a standby-sparing system. We propose a novel energy-aware mixed partitioning scheduling algorithm (EAMPSA). Earliest deadline first/dynamic deadline modification (EDF/DDM) scheduling scheme is used to ensure that the shared resources can be accessed in a mutual exclusive manner. Uniformly speed is used to the primary processor and the spare processor. In addition, we use the mixed mapping partitioning of primary and backup tasks method to map tasks. A novel method of mapping task is proposed i.e. the tasks which need to access to shared resources are mapped into the primary processor and the tasks which have no resource requirements are mapped into the spare processor. Furthermore, DVS and DPM techniques are used for both primary and backup tasks to save energy. The experimental results show that the EAMPSA algorithm consumes average 55.43% less energy than that of the SSPT algorithm.

1. Introductions

Energy management is an important object for real-time embedded system. The general energy management techniques are dynamic voltage scaling (DVS) [1–2] and dynamic power management (DPM) [3]. The DVS technique dynamically adjusts processor supply voltage and frequency based on the system workload to reduce power consumption. The DPM technique puts devices into a sleep mode to reduce power consumption when they are not in use. In fact, the feasibility of the system has been met at run-time even when the response time of task increases because the energy management techniques are used.

Other important objects for real-time embedded system are reliability and fault tolerance etc. The real-time embedded system may occur transient faults or permanent faults [4]. Transient faults are often induced by cosmic ray radiations and electromagnetic interference. In addition, energy management technique and the increase of component density of CMOS circuits significantly increase the vulnerability of systems to transient faults [5–6]. Time-redundancy techniques are commonly used to deal with transient faults. Available slack time is very important for time-redundancy techniques and it is used to construct recovery tasks to fault tolerance [7]. Permanent faults which lead to the unavailability of the system until they are repaired or replaced are often induced by manufacturing defects and circuit wear out. Hardware redundancy techniques are commonly used to deal with permanent faults. They are implemented by the extra processor which incurs considerable energy overhead. Duplex and Triple modular redundancy (TMR) [8] are two well-known hardware-redundancy techniques that clearly increase the energy consumption. Therefore, we can significantly improve the system reliability offered by hardware redundancy and we also take account of the energy consumption.

Energy and reliability are two important design dimensions in real-time system. Some research works [9–11, 25] have addressed both low energy-consumption and fault tolerance in hard real-time systems that are based on time redundancy. However, they focus on the time-redundancy to deal with transient faults and don't consider hardware redundancy. The multicore/multiprocessor systems which use additional processor units to deal with the co-management of energy and reliability become more appealing. With extra processors (hardware redundancy), the system can tolerate permanent faults through the extra processor. Standby-sparing systems are a particularly interesting framework for hardware redundancy [5–6, 12–15].

The first work addresses both energy and reliability in a standby-sparing system in [14]. In a standby-sparing system, there is a primary processor and a spare processor. The primary task is executed on the primary processor and the related backup task is scheduled on the spare processor. The primary processor can use both DVS and DPM

* Corresponding author at: College of Computer Science and Technology, Huaqiao University, XiaMen 361021, China.
E-mail address: zyw@hqu.edu.cn.

https://doi.org/10.1016/j.csi.2018.06.004
Received 30 November 2017; Received in revised form 13 June 2018; Accepted 18 June 2018

techniques to minimize the energy consumption of the system. The spare processor only uses the DPM technique to put the spare processor into a sleep mode and schedules the backup task as late as possible. Therefore, some works aim to minimize the overlap between the two copies of the same task on both processors to cancel the backup task execution on the spare processor when the related primary task completes its execution successfully on the primary processor [14–15]. However, these works are limited to aperiodic non-preemptive tasks. In fact, many significant real-time applications are preemptive and periodic. Haque et al. [12] proposed an energy-aware standby-sparing technique for periodic real-time applications, called SSPT algorithm, which the primary task and backup task are scheduled by the Earliest Deadline First (EDF) policy and Earliest Deadline Late (EDL) respectively. The SSPT algorithm is computationally expensive. In addition, it assumes that the tasks are independent and it schedules the backup task with the maximum processor speed. Some works focus on the fixed priority periodic task scheduling in a standby-sparing system [5–6]. The standby-sparing fixed-priority (SSFP) algorithm has been proposed to solve the problem of the energy and reliability in a fixed priority standby-sparing system [5]. The SSFP algorithm uses a dual-priority scheduling framework and a delayed promotion rule to dynamically postpone the execution of the backup task as much as possible when the earlier backup tasks are cancelled at runtime. Unlike [5], Moghaddas et al [6] proposed a reliability oriented energy-efficient scheduling method for static-priority workloads that employs both DVS and DPM in both primary and backup tasks. It uses concurrency and redundancy approach to schedule task. Note that, the above works assume that all tasks are independent i.e. the tasks don't need to access to shared resources during their execution.

To the best of our knowledge, there is no existing work that addresses how to effectively schedule periodic real-time tasks with shared resources by dynamic-priority scheme on a standby-sparing system to save energy with both DVS/DPM techniques while preserving the system original reliability with respect to transient faults and tolerating a single permanent fault. We focus on this problem and propose a novel energy-aware mixed partitioning scheduling algorithm (EAMPSA). In particular, the contributions of this work are summarized as follows:

First, the EDF/DDM scheduling scheme is used to ensure that the shared resources can be accessed in a mutual exclusive manner. The uniformly speed is used in the primary processor and the spare processor.

Second, we use the mixed mapping partitioning of primary and backup tasks method to map tasks. A novel method of mapping task is proposed i.e. the tasks which need to access to shared resources are mapped into the primary processor and the tasks which have no resource requirements are mapped into the spare processor. The DVS and DPM techniques are used for both primary and backup tasks to save energy while guaranteeing the deadline of tasks and the system's original reliability.

Finally, the proposed EAMPSA algorithm is evaluated through extensive simulations and the results show their effectiveness on energy savings.

The remainder of the paper is organized as follows: the model and problem descriptions are presented in Section 2. An energy-aware mixed partitioning scheduling algorithm is presented in Section 3. The simulation results and discussions are presented in Section 4 and conclusions and future work are presented in Section 5.

2. Models and problem description

2.1. Task model

We consider a set of periodic real-time task set $T = \{T_1, T_2, \ldots, T_n\}$ which is scheduled by an EDF scheme and a set of serially reusable and single unit resource $R = \{R_1, R_2, \ldots, R_m\}$. The reusable resources which can be a software object such as a data structure [16] must be accessed in a mutual exclusive manner. Each task T_i can be described by 3-tuples (e_i, r_i, p_i), where e_i is a worst case execution time (WCET) of T_i under the maximum processor speed $S_{\max}$ and r_i is a resource requirement of the task T_i and it is represented by an integer between 0 and m. If $r_i = 0$, it means that the task T_i doesn't use a resource during its execution. If $r_i \neq 0$, it means that the task T_i should use a resource R_{r_i} during its execution and other tasks which access to the resource R_{r_i} will be blocked. p_i is the period of T_i. In this paper, we assume that the relative deadline of T_i is equal to its period and that the task should access to at most one resource at a time. A set of periodic real-time tasks T is arranged in non-descending order according to the task's period i.e. $p_1 \leq p_2 \leq \cdots \leq p_n$. T_{ij} and $rt_{i,j}$ are denoted as the j^{th} job of the task T_i and the release time of the task T_i, respectively. Let u_i be the utilization of the task T_i and it can be expressed by $u_i = e_i/p_i$. Thus, the system utilization U_{tot} can be expressed by $U_{tot} = \sum_{i=1}^{n} u_i$. We assume that $U_{tot} < 1$ and that the WCET of T_i scales linearly with the processor speed i.e. the WCET of T_i is equal to $e_i \cdot S_{\max}/S_i$ with the processor speed S_i.

We use the EDF/DDM policy [16] to ensure that shared resources can be accessed in a mutual exclusive manner. The EDF/DDM policy is based on the earliest deadline first (EDF) scheme and it has two kinds of deadlines for job T_{ij}. The initial deadline ($ID_{i,j}$) is assigned when the job arrives. In addition, the execution deadline ($ED_{i,j}$) is assigned when the job begins to execute. Anyway, $ED_{i,j}$ is equal to $ID_{i,j}$ when the job arrives. The priority of jobs is determined by its execution deadline i.e. the smaller the execution deadline, the higher the priority. The initial deadline can be calculated as follows:

$$ID_{i,j} = rt_{i,j} + p_i \tag{1}$$

In addition, the execution deadline can be calculated as follows [16]:

$$ED_{i,j} = \min\{rt_{i,j} + p_i, \lfloor t_{i,j} \rfloor + 1 + MP_i\} \tag{2}$$

Where $t_{i,j}$ is the commences execution time of the job $T_{i,j}$, MP_i is the shortest period of the task that needs to access to a resource R_i and it can be expressed by $MP_i = \min_{1 \leq j \leq n} \{p_j | r_j = i\}$. Let $priority(T_{i,j})$ be the priority of the job $T_{i,j}$.

For reliability and fault tolerance goals in a standby-sparing system, each task T_i has a backup task B_i. B_i has a same time parameter as T_i. Let B_{ij} be the j^{th} job of the task B_i. The occurrence of transient faults can be detected at job completion time, an acceptance (or, consistency) test [4]. When a fault is detected, the backup job is executed to improve the system reliability. Otherwise, the backup job is immediately cancelled or terminated. The cost of running the acceptance test is assumed to be included in the WCET of jobs [12].

2.2. Power and energy model

Each processor can be operated in three modes: active mode, idle mode and sleep mode [5–6]. The tasks are executed in an active mode. When a processor doesn't execute the task, the processor may be in an idle mode or a sleep mode. Different processor modes consume different power and energy consumption.

2.2.1. Active mode

The power consumption model in the active mode can be followed as recent work [3, 5–6]. It consists of dynamic power and static power. The static power P_s is caused by leakage current of the system and it can be treated to 0 when the whole system is turned off. The dynamic power P_d consists of a frequency-dependent power P_{dep} and a frequency-independent power P_{ind}. P_{ind} is caused by modules such as memory and I/O devices and P_{dep} is dependent on the processor speed. In short, the power in an active mode P_{active} can be expressed as follows:

$$P_{active} = P_s + P_{ind} + P_{dep} = P_s + P_{ind} + C_{ef} \cdot S^m \tag{3}$$

Where C_{ef} is an effective switching capacitance, S is the processor speed. In this paper, we assume the processor can provide continuous