Scheduling Algorithms Based on Weakly Hard Real-Time Constraints

TU Gang (~), YANG FuMin (~H~) and LU YanSheng (~)
School of Computer Science, Huazhong University of Science and Technology, Wuhan 430074, P.R. China
E-mail: tugang@163.net
Received July 17, 2002; revised June 27, 2003.

Abstract The problem of scheduling weakly hard real-time tasks is addressed in this paper. The paper first analyzes the characters of μ-pattern and weakly hard real-time constraints, then, presents two scheduling algorithms, Meet Any Algorithm and Meet Row Algorithm, for weakly hard real-time systems. Different from traditional algorithms used to guarantee deadlines, Meet Any Algorithm and Meet Row Algorithm can guarantee both deadlines and constraints. Meet Any Algorithm and Meet Row Algorithm try to find out the probabilities of tasks breaking constraints and increase task's priority in advance, but not till the last moment. Simulation results show that these two algorithms are better than other scheduling algorithms dealing with constraints and can largely decrease worst-case computation time of real-time tasks.

1 Introduction

Bernat and Burns first introduced the definition of Weakly Hard Real-Time (WHRT) in [1]. In WHRT system, met or missed deadlines during consecutive periods must obey some constraints[1,2]. Bernat defined four generally used WHRT constraints[3]. These WHRT constraints and the representing method of real-time tasks can be easily applied to other real-time systems. Therefore, different types of real-time tasks can be described and studied in a united way, and it is valuable to design scheduling algorithms based on WHRT constraints.

There are some other scheduling algorithms used to guarantee constraints of consecutive periods, such as Static Promotions Selection (SPS) algorithm and Dynamic Promotions Selection (DPS) algorithm presented by Bernat in [4], Distance-based Priority (DBP) assignment scheme by Hamdaoui in [5], Dynamic Window-Constrained Scheduling (DWCS) algorithm by West in [6], and scheduling policy used by MELODY system[7]. Although these scheduling algorithms can greatly decrease the chances that tasks break constraints, they cannot fully guarantee constraints and efficiently decrease the worst-case computation time of real-time tasks.

Bi-Modal Scheduler (BMS) algorithm by Bernat[9] is designed for WHRT constraints. It is a two-mode scheduler. Tasks switch between normal mode and panic mode. Only when the system is under load, BMS can fully guarantee WHRT constraints. BMS algorithm must wait to the m-th invocation of one window to determine whether a task should switch into panic mode. But in some cases, this is too late. Moreover, when there are several tasks in panic mode and the conditions in [8] cannot be satisfied, tasks will inevitably break constraints.

In this paper, we first analyze the properties of weakly hard real-time constraints, then propose two algorithms, Meet Any Algorithm (MAA) and Meet Row Algorithm (MRA), based on the analysis. These two algorithms try to find out the probabilities of tasks breaking constraints, then increase task's priority in advance. MAA and MRA can increase task's priority even if there are not (m - 1) finished invocations; therefore, tasks have more chances to increase priority and avoid breaking constraints. Simulation results show that the MAA and MRA algorithms have better performance than DBP, BMS and MELODY and can significantly decrease the worst-case computation time of real-time tasks.

2 Specification of Weakly Hard Real-Time

The representing methods of deadlines introduced by Bernat in [1] can be applied to other real-time systems. The definition of μ-pattern and constraints can also be used to analyze other types of real-time tasks.
positive real numbers used to decide weights of which is used to guarantee WHRT constraints. 

To describe real-time task, a met deadline is represented with 1 and a missed deadline represented with 0. Deadlines that a task met or missed are denoted by 0,1 sequences, called $\mu$-pattern, denoted by $\omega$, $\omega \in \Sigma^*$, $\Sigma = \{0,1\}$. $L(\omega)$ is used to denote the length of $\omega$ and $\omega(i)$ ($1 \leq i \leq L(\omega)$) is used to denote the $i$-th element of $\omega$. For example, for task set $\Pi = \{T_1, T_2, \ldots, T_n\}$, $T_2 = (15, 15, 6, 0, \lambda)$, $T_3 = (25, 25, 2, 0, \lambda)$, $T_4 = (30, 30, 2, 0, \lambda)$ scheduled by RM algorithm, the $\mu$-pattern of $T_2$ is $\omega_2 = 0101010101 \ldots$.

### 2.2 Weakly Hard Real-Time Constraints

There are four generally used WHRT constraints for one window with $m$ consecutive periods: $\text{Meet any } n \text{ in } m$, denoted by $({n \choose m}$, $\text{Miss any } n \text{ in } m$, denoted by $({n \choose m}$, $\text{Meet row } n \text{ in } m$, denoted by $({n \choose m}$, $\text{Miss row } n \text{ in } m$, denoted by $({n \choose m}$.

### 3 Scheduling Algorithms Based on WHRT Constraints

Scheduling algorithms for WHRT systems should guarantee not only deadlines but also WHRT constraints, therefore, the priority formulas of the algorithms presented in this paper are composed of two parts: real-time formula and constraint formula. The main difference between priority formulas of MAA and MRA is their constraint formula. The priority formula of task $T$ for MAA and MRA is given by:

$$\text{Priority}(T) = \alpha \times T(T) + \beta \times C(T) \quad (1)$$

Here $T(T)$ is real-time formula, which is used to guarantee task’s deadlines. $T(T)$ can be replaced by priority formula of other scheduling algorithms, such as RM, EDF. $C(T)$ is constraint formula, which is used to guarantee WHRT constraints. $C(T)$ is different in MAA and MRA. $\alpha$ and $\beta$ are positive real numbers used to decide weights of $T(T)$ and $C(T)$. Here, larger value of $\text{Priority}(T)$ means higher priority. $C(T)$ is the main part of $\text{Priority}(T)$, therefore, in this section we mainly consider the problem of guaranteeing WHRT constraints and introduce the $C(T)$ formula.

### 3.1 MAA

SPS and DPS algorithms introduced by Bernat and Burns and DWCS algorithm by West only care about the number of met deadlines in the last $(m-1)$ consecutive periods, that is, while assigning priority of tasks they only calculate the number of 1’s in $\mu$-pattern $\omega L(\omega) - m - 1, m - 1$ is used to denote the arriving invocation). This method is simple but not sufficient. From the analysis below we know that the priority of tasks is also determined by the numbers of 1’s in sub-$\mu$-pattern of $\omega L(\omega) - m - 1, m - 1$. DBP introduced by Hamdaoui and Ramanathan is based on $(m,k)$-firm constraint, which is the same as “meet any $n$ in $m$” WHRT constraint. Using state machine, DBP has a complexity of $O(2^m)$ and needs much memory. The MAA algorithm presented here is based on the analysis of properties of sub-$\mu$-pattern of $\omega L(\omega) - m - 1, m - 1$. It uses the definition of “harder” and has a complexity of $O(m)$.

$$\omega(L(\omega)) \text{ belongs to } m \text{ consecutive windows: } \omega L(\omega) - m - 1, m - 1, \omega L(\omega) - m + 2, m - 1, \ldots, \omega L(\omega) - m, m. \text{ The value of } \omega(L(\omega)) \text{ may influence whether the constraint can be satisfied in any of these } m \text{ consecutive windows. Therefore, the priority of task should be determined by the importance of } \omega(L(\omega)) \text{ in these } m \text{ consecutive windows. For } \omega(L(\omega)), \text{ only } \mu \text{-pattern } \omega L(\omega) - m - 1, m - 1 \text{ has } (m-1) \text{ finished invocations which can be used to determine the priority. However, } \omega L(\omega) - m + 2, m - 1, \ldots, \omega L(\omega) - m, m \text{ have not enough finished invocations: } \omega L(\omega) - m + 2, m - 1 \text{ has } (m-2) \text{ finished invocations and } \omega L(\omega) - m, m \text{ has zero. In [3], Bernat introduced a method to find out the nether max “harder” constraints of } {n \choose m} \text{ and these nether max constraints are for } (m-2), \ldots, 1 \text{ windows. We can use these constraints to determine the importance of } \omega(L(\omega)) \text{ in } \omega L(\omega) - m + 2, m - 1, \ldots, \omega L(\omega) - m, m. \text{ If } \mu \text{-pattern breaks one of these constraints, it must break } {n \choose m}, \text{ therefore, the priority of tasks can be determined by these nether max “harder” constraints.}

**Theorem 1.** The number of met deadlines of the last $(m-1)$ consecutive periods can determine the priority of tasks that satisfy $({n \choose m}$ constraint.

**Proof.** There is close relationship between $\mu$-patterns $\omega^{i-1} m$ and $\omega^m$: if $\omega^{i-1} m$ $(1 \leq i \leq L(\omega) - m + 1, m \geq 1)$ satisfies $({n \choose m}$ constraint and $\omega(i + m - 1) = 1$. $\omega^m$ will also satisfy the con-