Journal of Systems Architecture 79 (2017) 31–44

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

An efficient approach for the multiprocessor non-preemptive strictly periodic task scheduling problem

Omar Kermia

CDTA, Algiers, Algeria

ARTICLE INFO

Article history:
Received 1 August 2016
Revised 18 April 2017
Accepted 15 July 2017
Available online 19 July 2017

Keywords:
Real-time systems
Schedulability analysis
Strictly periodic tasks
Multiprocessor platform

ABSTRACT

Strict periodicity constraint is of great importance since it concerns some hard real-time systems where missing deadlines leads to catastrophic situations. However, the problem of schedulability analysis for non-preemptive strictly periodic tasks on a multiprocessor platform is even more intractable than the one with the common periodicity. In order to implement such systems, designers need effective tools based on fast and near-optimal solutions.

This paper presents a schedulability analysis which results mainly in a, two versions, task assignment and start-time calculation algorithm. The first one targets the harmonic task periods case while the second one targets the non-harmonic task periods case. Each version is based on a sufficient uniprocessor schedulability test. In addition, for the non-harmonic case which is the most intractable, the uniprocessor sufficient schedulability test uses the strictly periodic task utilization factor. This factor stands for the fraction of time spent to execute a task while its strict periodicity and the ones of the already scheduled tasks are met. As a result, an efficient and easily implementable scheduling algorithm is proposed which begins by assigning tasks to processors then attributes a start-time to every task in such a way that strict periodicity and deadline constraints are met. The effectiveness of the proposed scheduling algorithm, in both versions, has been shown by a performance evaluation and comparisons with an optimal and a similar suboptimal solution.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

Hard real-time problematic is to maintain temporal and functional achievement of system execution. Hard real-time scheduling has been concerned with providing guarantees for temporal feasibility of task execution whatever the situations. A scheduling algorithm is defined as a set of rules defining the execution of tasks at system run-time. A schedulability analysis, which determines whether a set of tasks with parameters describing their temporal behavior will meet their temporal constraints if executed at run-time according to rules dictated by the scheduling algorithm. The result of such a test is typically a *yes* or a *no* answer indicating whether a solution will be found or not. These schemes and tests demand precise assumptions about task properties, which hold for the entire system lifetime. Generally, we talk about system failure when timing constraints are not met. For this very reason, in this kind of system, timing constraints must be guaranteed to be met. In addition, real-time systems need to be predictable in order to guarantee the timeliness of their behavior. Time-predictability is exactly one of the preconditions necessary to verify the correct operation of a real-time system [1]. While satisfying the timing constraints of the system, it is also desirable that the system achieve a high level of processor utilization. Only processing requirements are significant; memory, I/O, and other resource requirements are negligible.

In some hard real-time systems some tasks are repeated according to a strict periodicity [2,3] which is more constrained than the most studied period tasks model initially proposed by Liu and Layland [4]. Therefore, available results addressing the non-preemptive periodic scheduling such as in [5–7] are not suitable for strict periodicity case.

Strictly periodic tasks could be sensors/actuators for which the relevancy of information they use is linked to the accuracy of the repetition. Hence, these tasks must be executed following a strict periodicity [8]. In real-time control systems, one of the consequences of strict periodicity is that the separation between consecutive task instance starting times, called sampling interval, becomes constant. This stability is synonymous with the control of jitters which are delays between tasks ready-times and when they start their execution [9–12]. The fact that we are dealing with pre-

E-mail addresses: omar.kermia@gmail.com, okermia@cdta.dz

http://dx.doi.org/10.1016/j.sysarc.2017.07.005
1383-7621/© 2017 Elsevier B.V. All rights reserved.

emptive scheduling does not affect the latency performance considerably (compared to a non-preemptive scheduling) since there exist some methods allowing minimizing latency which leads to robust preemptive real-time control systems [13–15]. Also, the strict periodicity is a part of the constraints in Mixed Criticality System such as with TTEthernet protocol [16] or IMA "Integrated Modular Avionic" based systems [17].

In this work, we are interested in an effective off-line suboptimal scheduling of a strictly periodic real-time task set on a number of processors. This requires the study of the schedulability of such tasks on a single processor in order to find how to determine whether a task set is schedulable on a processor in a reasonable amount of time. Then, based on the study results, the goal is to develop a scheduling algorithm that returns the start-time execution of every task and the processor on which it will be executed in such a way that strict periodicity of all tasks is met. To this end, we propose to distinguish between the harmonic and the non-harmonic period cases seeing that the harmonic one stands for a less intractable problem. This allows getting a harmonic task scheduling algorithm with less complexity than that of the general case. For the non-harmonic case, we propose to assign tasks and give them start-times within two successive processes. Indeed, for scheduling strictly periodic tasks, we believe that it is more profitable to begin by finding a feasible assignment for tasks while minimizing the number of used processors and then attribute the suitable start-times. Proceeding in this way reduces the complexity of the scheduling algorithm but, nevertheless, requires a schedulability condition that does not involve start-time parameter. In addition, the proposed tasks assignment seeks to divide the task set into feasible subsets; and tasks within the same subset are assigned to the same processor.

The first contribution of this paper is a sufficient schedulability condition followed by a polynomial multiprocessor tasks assignment and start-time calculation algorithm for harmonic tasks. Then, to address the non-harmonic case, the strictly periodic task utilization factor is introduced. It stands for the sum of fractions of time spent to execute a task in such a way that its strict periodicity and the ones of the already scheduled tasks are met. After that, based on the proposed factor, a polynomial sufficient schedulability condition for the uniprocessor case is introduced such that: (i) it does not involve tasks start-time and (ii) it has any restriction about tasks period values. Once the schedulability of a task subset is assessed, we propose to compute tasks start-times that fulfill both strict periodicities and deadlines constraints. The scheduling algorithm, for non-harmonic tasks, is of pseudo-polynomial complexity; it returns a minimized number of required processors and the assignment of every task (with a start-time) to one of these processors. In this way, every task's start-time is computed only once without changing any already computed task start-time. Furthermore, a performance evaluation including comparisons with both optimal method and an efficient suboptimal solution is proposed.

This paper is organized as follows. After a review of the related work in Section 2, the scheduling problem is formally presented in Section 3. Sections 4.1 and 4.2 present the performed schedulability analysis and the scheduling algorithm for harmonic and non-harmonic tasks respectively. Section 5 gives some experimentations and results. Section 6 finally concludes and gives some perspectives on future work.

2. Related work

In [3], Korst et al. showed that the problem of the non-preemptively scheduling strictly periodic tasks is NP complete in the strong sense in the case of a single processor.

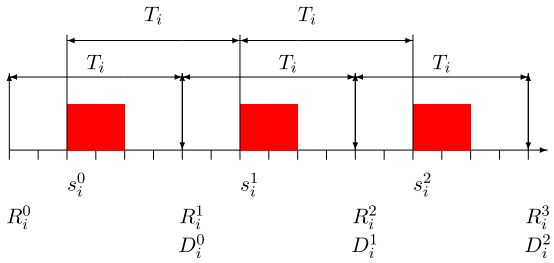

Fig. 1. Model for a non-preemptive strictly periodic task.

In addition, Kermia and Sorel proposed in [18] a necessary schedulability condition which was proven to be very restrictive [19]. Later, Eisenbrand et al. [20] addressed the special case of harmonic periods, i.e., every period divides all other periods of greater value. They showed that there exists a 2-approximation for the minimization of the processors number. Then, Sheikh et al. [17] addressed the optimal solution by proposing a best-response algorithm based on the game theoretic approach. This work has been improved by Pira and Artigues [21] by proposing a new method to compute the best response, and a propagation mechanism for non-overlapping constraints. More recently, Chen et al. [22] presented an efficient suboptimal task assignment algorithm which is based on a uniprocessor schedulability condition which determines whether a new task can be scheduled on a processor without changing the start-times of the already scheduled tasks and identifies all valid start-times for the new task if it is schedulable. This uniprocessor schedulability condition takes into account only tasks whose periods are equal to, or multiples of, either one of the periods or the Greatest Common Divisor of all periods of the already scheduled tasks. Zhang et al. [23] tackled the uniprocessor case by proposing a sufficient test condition to check the feasibility of a given task set, and returns their start-times.

3. Model

We deal with systems of non-preemptive strictly periodic real-time tasks which are considered as synchronous since they are all first activated at the same time $t = 0$. Hence, a system consists of a set of n concrete strictly periodic tasks denoted by Γ ($|\Gamma| = n$). A task i denoted by $\tau_i \in \Gamma$ is an infinite sequence of jobs denoted by $(\tau_i^j, j \geq 0)$ where each job is characterized by: an execution start-time s_i^j, a strict period T_i, an activation time R_i^j (where $\forall\ i, R_i^0 = 0$), a relative deadline denoted by D_i^j equal to its period ($D_i^j = T_i$) and a worst case execution time C_i. As periods are strict then s_i^j, the start-time of the jth job of τ_i, is given by: $\boldsymbol{s_i^j = s_i^0 + j \cdot T_i}$. The hardware platform is an m identical processors.

Fig. 1 shows an example of a non-preemptive strictly periodic task scheduling. In addition, we assume time is discrete and clock ticks are indexed by the natural numbers. Hence, saying that a job τ_i^j starts at time s_i^j means that it starts its execution at the start of the interval $[s_i^j, s_i^j + 1)$. All tasks are considered as independent meaning that there are no precedence constraints between tasks.

4. Schedulability analysis

We propose a sufficient schedulability condition and a scheduling algorithm for harmonic and non-harmonic tasks respectively.

4.1. Harmonic tasks

4.1.1. Uniprocessor case

Let us consider a set of harmonic periods tasks $\Gamma = \{\tau_1, .., \tau_n\}$. Without any loss of generality, harmonic periodic tasks are as-

Download English Version:

https://daneshyari.com/en/article/4956221

Download Persian Version:

https://daneshyari.com/article/4956221

Daneshyari.com