A 2.79 COMPETITIVE ON-LINE ALGORITHM FOR TWO PROCESSOR REAL-TIME SYSTEMS WITH UNIFORM VALUE DENSITY

YANG QIFAN

Abstract. We consider the problem of competitive on-line scheduling in two processor real-time systems. In our model all tasks have common value density. Each task has a release time, an execution time and a deadline. The scheduler is given no information about a task until it is released. And the value will be achieved if and only if the task is completed by its deadline. Moreover, we suppose that migration is not allowed. The goal of the scheduler is to obtain as much value as possible. In this paper we show that the competitive multiplier of Safe-Risky-fixed Algorithm in [4] is really 3 and presents a modified algorithm (Safe-Risky-unfixed Algorithm) that achieves a competitive multiplier of 2.79.

1. Introduction

We consider the problem of two processor real-time systems. In our model every task has a release time, an execution time and a deadline. Each task must be executed between its release time and deadline and a value equal to the task’s execution time is obtained, otherwise the system gets no value from the task. We also suppose that a task can be executed on any processor. To preempt an executed task is allowed and preemption within a processor takes no time but migration is not allowed. It means that once a task is scheduled on a processor it could not be moved to another one any more. Moreover, the scheduler is given no information about a task until it is released and the goal is to get as much value as possible.

To quantify an on-line algorithm, in a general way, we compare it with a so-called clair...
voyant algorithm which knows all information about the tasks ahead and can schedule tasks skillfully to achieve a best result. An algorithm is said to have a competitive multiplier of \( a \) if it guarantees that the clairvoyant algorithm can not achieve value more than \( a \) times the value achieved by it.

The problem of competitive on-line scheduling for the real-time systems has been motivated and considered by several authors in recent few years. For uniprocessor environments, Baruah, S. et al. [1,2] showed a lower bound of \((1 + \sqrt{k})^2\) where \( k \) is the so-called importance ratio\(^*\). Wang F. and Mao D. [6] presented an algorithm that achieves this bound with \( k = 1 \) (uniform value density). Koren G. and Shasha D. [5] presented an optimal algorithm (D\(^{opt}\) Algorithm) that achieves the bound \((1 + \sqrt{k})^2\) for all \( k \geq 1 \). For multiprocessor environments, Dertouzos M. and Mok A. [3] showed that no optimal algorithm exists even when the system is underloaded. Wang F. and Mao D. [1,6] showed that no on-line algorithm can achieve a competitive multiplier of \( a \) which is smaller than 2. They presented an algorithm that achieves bound 2, assuming uniform value density and that all task are urgent (no slack time). Koren G. and Shasha D. [4] considered the problem of competitive on-line scheduling for multiprocessor real-time systems. For two processor systems they presented an algorithm (Safe-Risky Algorithm) that achieves a competitive multiplier of 2, assuming that tasks may have slack time but migration is allowed. They also presented an algorithm (Safe-Risky-fixed Algorithm) that can be used when migration is not allowed and showed that the competitive multiplier of Safe-Risky-fixed Algorithm is not larger than 3.

In this paper we show that the competitive multiplier of Safe-Risky-fixed Algorithm is really 3. And we present a modified algorithm (Safe-Risky-unfixed Algorithm) that achieves a competitive multiplier of 2.79.

2. Review of Safe-Risky-fixed Algorithm (SR-fixed)

To begin with, we describe the Safe-Risky-fixed Algorithm presented by Koren G. and Shasha D. [5]. For simplicity we will denote the execution time and deadline of a task \( T \) by \( C_T \) and \( d_T \) respectively. SR-fixed algorithm designates one processor as the Safe Processor (SP) and another one as Risky Processor (RP). Corresponding to SP, there is a set \( Q \) which contains all preempted tasks. And there is another set \( W \) which contains all waiting tasks. Besides, we will call the tasks executed currently on SP and RP \( T_{current\ (SP)} \) and \( T_{current\ (RP)} \) respectively.

\(^*\) A task's value density is the ratio of its value to its execution time. The importance ratio of a set of tasks is the ratio of the largest value density to the smallest value density.