A Proposal of Real-Time Scheduling Algorithm based on RMZL and Schedulability Analysis

Ken YANAI\textsuperscript{a,*}, Myungryun YOO\textsuperscript{a}, Takanori YOKOYAMA\textsuperscript{a}

\textsuperscript{a}Tokyo City University, 1-28-1 Tamazutsumi, Setagaya-ku, Tokyo 158-8557, Japan

Abstract

Recently, multiprocessor platform is generally used in embedded real time systems. The optimal real time scheduling algorithms for multiprocessor are demanded. Several algorithms based on RM are proposed. In this study, we propose RMZLPD based on RMZL applied zero-laxity rule to RM. RMZLPD can realize high parallelism. Through simulation, RMZLPD has shown the high schedule success ratio. The schedulability of proposed algorithm also is shown by response time analysis.

© 2013 The Authors. Published by Elsevier B.V.
Selection and peer-review under responsibility of the Program Committee of IES2013

Keywords: real-time scheduling algorithm, RMZL, RMZLPD, schedulability

1. Introduction

Real time systems are characterized by computational activities with timing constraints. Timing constraints in real time applications are predominantly soft in that deadlines may be missed as long as the long run fraction of the processing time allocated to each task in the application is in accordance with its utilization. A system design that can guarantee that deadline misses, if any, are bounded by constant amounts is sufficient to provide guarantees on long term processor shares. Hence, scheduling methods that ensure bounded deadline misses and that can be applied when other methods cannot are of considerable value and interest\textsuperscript{1}.

Multiprocessor scheduling are usually categorized into two paradigms: global scheduling, in which each task can execute on any available processor at runtime, and partitioned scheduling in which each tasks is assigned to a processor beforehand, and at runtime each task can only execute on this particular processor. Partitioned scheduling enjoys relatively easier design and analysis. On the other hand, global scheduling on average utilizes computing

\* Corresponding author. Tel.:+81-3-5707-0104 (Ext. 3737).
E-mail address: g1281539@tcu.ac.jp.
resource better, and is more robust in the presence of timing errors\(^2\). Global scheduling algorithms are based on widely optimal uniprocessor scheduling algorithms like RM (Rate Monotonic) and EDF (Earliest Deadline First) by Dhall\(^3\). However, these algorithms aren’t optimal on multiprocessor systems. Although many scheduling algorithms based on RM or EDF are proposed, optimal algorithms aren’t established yet. Compared to EDF, RM is easy to implement and tiny jitter. Thus, in this paper, we focus on global scheduling algorithm based on RM.

Takeda et al. proposed RMZL (Rate Monotonic until zero laxity) based on RM\(^4\) and Nishigaki et al. proposed LP-RMZL (Limited Preemptive-RMZL) based RMZL\(^5\). These algorithms show higher schedule success ratio than that of RM. However, there are still reducible deadline miss.

In this paper, we propose a new scheduling algorithm, called RMZLPD (RMZL with Pseudo Deadline). The proposed algorithm dominates global RM scheduling, RMZL and LP-RMZL. Also, we analyse schedulability of RMZLPD using Response Time Analysis (RTA).

The rest of the paper is organized as follows: In Section 2, we explain system model. In Section 3, global RM, RMZL and LP-RMZL are explained more detail. Section 4 introduces proposed RMZLPD. In section 5, a schedulability of RMZLPD is analysed. Then, the experimental results are illustrated and analysed in Section 6. Finally, Section 7 provides discussion and suggestions for further work on this problem.

2. System Model

The notation described in this Section. We consider a set \( \tau \) of \( n \) periodic tasks to be scheduled on \( m \) symmetric processors using a global algorithm.

Each task \( \tau_k = (C_k, T_k) \in \tau \) is characterized by a worst-case computation time \( C_k \), a period \( T_k \). The utilization of a task is defined as \( \mu_k = C_k/T_k \). The system utilization is defined as \( \mu = \sum_{k=1}^{n} \mu_k/m \). A task \( \tau_k \) is a sequence of jobs \( J_k \), where each job is characterized by an arrival time \( r_{jk} \) and a finish time \( f_{jk} \). Moreover, each job has an absolute deadline \( d_{jk} = r_{jk} + T_k \). The laxity of a job at time \( t \) is defined as \( L_{jk} = d_{jk} - t - C_{jk}(t) \), where, \( C_{jk}(t) \) is a remaining execution time of job \( J_{jk} \) at time \( t \).

3. Related Works

3.1. Global RM (Rate Monotonic)

Global RM is preemptive fixed priority scheduling algorithm. Tasks with higher request rates will have higher priorities in global RM. But, owing to low schedulability, it is known that global RM which is applied for multiprocessor platforms is not optimal scheduling algorithm\(^3\).

3.2. RMZL (Rate Monotonic until zero laxity)

![Fig. 1: Example of RM schedule and RMZL schedule](image)

Fig. 1: Example of RM schedule and RMZL schedule
RMZL is based on global RM. Under RMZL, jobs are scheduled according to the fixed priority of their associated task, until a situation is reached where the remaining execution time of a job is equal to the time to its deadline. Such a job has zero laxity and will miss its deadline unless it executes continually until completion.

RMZL gives the highest priority to such zero-laxity jobs. The schedules produced by RMZL and global RM scheduling are identical until the latter fails to execute a task with zero laxity. Such a task will subsequently miss its deadline. Hence RMZL dominates global RM scheduling, in the sense that all priority ordered task sets that are schedulable according to global RM scheduling are also schedulable according to RMZL. Figure 1 shows the scheduling example of global RM and RMZL, when three periodic tasks, \( r_1 = r_2 = r_3 = (2, 3) \) are submitted on two processors. As shown in the figure 1, \( r_3 \) misses a deadline at time 3 in global RM. On the other hand, all the three tasks are successfully scheduled by RMZL, since the priority of \( r_3 \) is promoted to the top at time 1 due to zero laxity.

3.3. LP-RMZL (Limited Preemptive-RMZL)

LP-RMZL is based on RMZL. Under LP-RMZL, running jobs are not preempted by higher priority tasks except for zero-laxity tasks. Compared to RMZL, LP-RMZL reduced preemption and improved success ratio and schedulability. Figure 2 shows the scheduling example of RMZL and LP-RMZL, when four periodic tasks, \( r_1 = r_2 = (1, 2), r_3 = (1, 4) \) and \( r_4 = (6, 8) \) are submitted on two processors. As shown in the figure 2, \( r_4 \) misses a deadline at time 8 in RMZL. On the other hand, all the four tasks are successfully scheduled by LP-RMZL, since the \( r_4 \) is not preempted at time 2, 4, and 6.

4. Related WorksRMZLPD(RMZL with Pseudo Deadline)

We propose RMZLPD added pseudo deadline to RMZL. Under RMZL, jobs are scheduled according to RMZL, until a situation is reached where the remaining pseudo execution time of a job is equal to the time to its pseudo deadline. Such a job has pseudo zero laxity and will miss its pseudo deadline unless it executes continually until its pseudo deadline.

RMZLPD gives the semi highest priority to such pseudo zero-laxity jobs until its pseudo deadline. Pseudo Deadline is set on the half deadline. Pseudo execution time also is set on the half execution time. Figure 3 shows the scheduling example of LP-RMZL and RMZLPD, when five periodic tasks, \( r_1 = r_2 = r_3 = (1, 4) \) and \( r_4 = r_5 = (6, 12) \) are submitted on two processors. As shown in the figure 3, \( r_5 \) misses a deadline at time 3 in LP-RMZL. On the other hand, all the five tasks are successfully scheduled by RMZLPD, since \( r_5 \) has the semi highest priority at time 5 due to pseudo zero laxity.
5. Schedulability

In this section, we derive sufficient schedulability test for RMZLPD with Response Time Analysis (RTA). 6

5.1. Interference and Workload

To analyze the schedulability, we define two parameters: the interference and the workload.

**Interference** The interference \(I_k(a, b)\) on \(\tau_i\) over an interval \([a, b]\) is the cumulative length of all intervals in which \(\tau_i\) is backlogged but cannot be scheduled on any processor due to the contemporary execution of \(m\) higher priority tasks. We also define the interference \(I_k^1(a, b)\) of a task \(\tau_i\) on a task \(\tau_k\) over an interval \([a, b]\) as the cumulative length of all intervals in which \(\tau_k\) is backlogged but cannot be scheduled on any processor, while \(\tau_i\) is executing.

**Workload** The workload \(W_i(a, b)\) of a task \(\tau_i\) in an interval \([a, b]\) represents the amount of computation that the task requires in \([a, b]\) on a given situation.

As for interference, the following lemma 1 is showed.

**Lemma 1** For any global scheduling algorithm it is:

\[
I_k(a, b) \geq \lambda \Leftrightarrow \sum_{i=k}^{m} \min\left(I_k^1(a, b), \lambda\right) \geq \lambda \geq m \lambda
\]  
(1)

The flow of an algorithm analysis is shown below.

Let \(J_k^*\) be the job of \(\tau_i\) with maximum interference. The upper bound of \(I_k\), \(I_k^{ub}\), can be calculated over an interval \([r_k^*, r_k^* + R_k^{ub}]\) from arrival time to response time of \(J_k^*\). Also, the upper bound of response time of \(\tau_i\), \(R_k^{ub}\), is derived from \(I_k^{ub}\) and the execution time of \(\tau_i\). And then, we can calculate the lower bound of laxity of \(\tau_i\), \(L_k^{lb}\) from the following equation (2).

\[
L_k^{lb} = T_i - R_k^{ub}
\]  
(2)

The schedulability of an algorithm is analyzed using \(L_k^{lb}\).

In here, we have lemma 2 as for \(I_k^{ub}\).

**Lemma 2** \(I_k^1(a, b)\) is always smaller than \(W_i(a, b)\). Therefore, \(I_k^{ub}\) can be calculated by calculating \(W_i^{ub}\).
5.2. Schedulability of RMZLPD

Under RMZLPD scheduling, if the laxity or the pseudo laxity of a job reaches zero then it is given the highest or the semi highest priority. Therefore, $\tau_k$ is interfered with not only higher static priority tasks but also lower static priority tasks. If $\tau_k$ is higher static priority than $\tau_i$, workload of $\tau_i$ over an interval $[r^*_k, r^*_k + R^{ab}_k]$ is represented in figure 4.

Thus,

$$W^{ia}(R^{an}_k) = C_i + \min \left( \frac{C_i}{2}, \max \left( 0, R^{an}_k - \frac{T_i}{2} - \frac{C_i}{2} \right) \right) \text{ if } (i > k)$$

If $\tau_k$ is lower static priority than $\tau_i$, workload of $\tau_i$ over an interval $[r^*_k, r^*_k + R^{ab}_k]$ is represented in figure 5.

Thus,

$$W^{ia}(R^{an}_k) = n_i(R^{an}_k)C_i + \min \left( C_i, R^{an}_k + I_k - C_i - L^p_k - n_i(R^{an}_k)T_i \right) \text{ if } (i < k)$$

where $n_i(R^{an}_k)$ is the maximum number of jobs of task $\tau_i$ that contribute all of their execution time in the interval:

$$n_i(R^{an}_k) = \left| \frac{R^{an}_k + I_k - C_i - L^p_k}{T_i} \right|$$

**Theorem 1 (RTA for RMZLPD)** An upper bound on the response time of a task $\tau_k$ in a multiprocessor system scheduled with RMZLPD can be derived by the fixed point iteration on the value $R^{an}_k$ of the following expression, starting with $R^{an}_k = C_k$:

$$R^{an}_k \leftarrow C_k + \left( \frac{1}{m} \sum_{i=1}^{m} \hat{f}_k(R^{an}_k) \right)$$

with

$$\hat{f}_k(R^{an}_k) = \min(W^{ia}(R^{an}_k), R^{an}_k - C_k + 1).$$

**Theorem 2 (Schedulability for RMZLPD)** A periodic task system $\mathcal{F} = \{ \tau_1, \tau_2, \ldots, \tau_n \}$ is schedulable by RMZLPD on $m$ symmetrical processors unless the following inequality holds for at least $m + 1$ different tasks $\tau_0$ and it holds strictly($<$)for at least one of them:
6. Experimental Evaluation

In order to see how well the RMZLPD algorithm and the above schedulability test perform, a series of experiments were conducted with the same simulation environment and we assumed maximum amount of processor is 16. Figure 6 shows the result of simulation about success ratio, when 1,000 task set (system utilization 0.3 ~ 1.0) are submitted on four processors. For comparison, three scheduling algorithm mentioned previously were tested. It indicates that RMZLPD is superior to other algorithms over an interval [30%, 100%].

![Figure 6](image1.png)

Figure 6. The result of simulation

![Figure 7](image2.png)

Figure 7. The result of schedulability.

Figure 7 shows the result of schedulability, when 1,000 task set (system utilization 0.3 ~ 1.0) are submitted on four processors. RMZLPD has two priority promotion chances. Therefore, RMZLPD is more interfered with static lower priority tasks than RMZL and LP-RMZL.

7. Conclusions

A new tasks scheduling algorithm on real time multiprocessor system is proposed in this paper. The schedulability of the proposed algorithm, RMZLPD, is analysed using RTA. RMZLPD has high schedule success ratio and can realize high parallelism. From the numerical results, the results of the proposed RMZLPD are better than that of other algorithms. However, there are a gap between the schedulability and the success ratio. This determines the next step of our study. We plan to analysis the schedulability with another method other than RTA.

Acknowledgements

This work is supported in part by KAKENHI (24500046).

References