

실시간 멀티프로세서 시스템에서의 태스크 스케줄을 위한 L-RE 좌표 알고리즘

황 월*, 김 용수*

L-RE Coordinates Algorithm for Task Scheduling in Real-time Multiprocessor System

Yue Huang*, Yong Soo Kim*

요 약

태스크 스케줄링은 여러 개의 수행 가능한 태스크 중에서 하나의 태스크를 선정하여 프로세서에 할당하는 중요한 역할을 한다. 실시간 시스템에서 경성 실시간 태스크를 적시에 스케줄링하지 못하면 시스템이 다운되는 최악의 상태가 발생하므로 실시간 시스템은 효율성, 자원 및 속도 등의 문제 외에도 시간 제약도 진지하게 고려해야 한다. 본 논문에서는 L-RE 좌표를 이용하여 실시간 다중프로세서 시스템에서의 새로운 우선순위 기준 스케줄링 알고리즘을 제안한다. L-RE 좌표 알고리즘은 태스크의 스케줄링 효율을 높이기 위해 고안 되었으며 우선순위를 할당하는데 데드라인과 함께 유희시간을 고려하고 있다. 시뮬레이션 결과는 LR-E 알고리즘이 EDF보다 스케줄의 원활성을 높이고 또 LLF보다는 문맥교환 수를 줄일 수 있음을 보여준다.

Abstract

Task scheduling is an essential part of any computer system for allocating tasks to a processor of the system among various competitors. As we know, in real-time system, the failure of scheduling a hard real-time task may lead to disastrous consequence. Besides efficiency, resource and speed, real-time system has to take time constraint in serious consideration. This paper proposes a priority-driven scheduling algorithm for real-time multiprocessor system, which is called L-RE coordinates algorithm. L-RE coordinates is a new way of describing the task scheduling problem. In the algorithm, we take both deadline and laxity into consideration for allocating the priority. The simulation result shows that the new algorithm is viable and performance better than EDF and LLF algorithm on schedulability and context switch respectively.

▶ Keyword : L-RE 좌표, 스케줄링 알고리즘, 실시간 시스템, 다중프로세서, L-RE Coordinates, scheduling algorithm, realtime system, multiprocessor

• 제1저자 : 황월 • 교신저자 : 김용수
• 접수일 : 2007.5.14, 심사일 : 2007.7.17, 심사완료일 : 2007. 7.23
* 경원대학교 소프트웨어대학

3.3 L-RE coordinates

In the previous studies on task scheduling algorithms, researchers used vectors to describe the model and exploited the numeration relationship among the parameters in the vector to calculate the priority for every task at an instance in time. Here we suggest a new L-RE coordinates to improve the real-time scheduling efficiency. The coordinates can represent all the properties of a task and also reveal some relationships among the properties which are not found easily. Figure 1 describes a L-RE coordinates with five tasks. Each task is represented as a node in the figure.

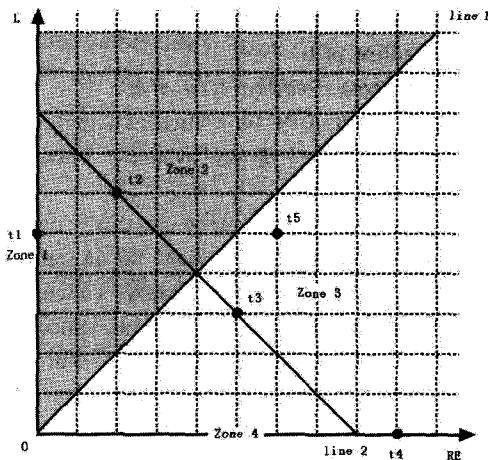


그림 1. L-RE 좌표
Fig. 1 L-RE coordinates

In the coordinates, x-axis indicates a task's remaining execution time r_i and y-axis indicates its laxity l_i . The metric unit is the length of the time unit in real-time system. The tangent value which is defined the ratio between the laxity and remaining time as in Formula 1 is used to describe the node's state.

$$\text{Tangent value } T_i = \frac{l_i}{r_i} \dots\dots\dots (1)$$

As the Figure 1 shows, the L-RE coordinates space can be divided into four zones:

- Zone 1 is attached to the y-axis. Nodes in this

zone, where $l_i > 0, r_i = 0$, have a infinite tangent value. It denotes that the tasks in this zone are successfully finished, like t_1 in Figure 1.

- Zone 2 is the space between y-axis and the line 1. Here line 1 is fixed with gradient 1 and origin (0,0). Nodes in this zone, where $l_i > r_i$, have the tangent value not less than 1, like t_2 . For the task's laxity is larger than remaining execution time, the nodes are not emergent and can still wait for a while.
- Zone 3 is the space between line 1 and x-axis. Nodes in this zone, where $l_i < r_i$, are with the tangent value less than 1, like t_3 and t_5 . For the task's laxity is less than remaining execution time, current time is near the latest start time. So the tasks should be elevated to higher priorities.
- Zone 4 is attached to the x-axis. Nodes in this zone, where $l_i = 0, r_i > 0$, are with the tangent value of 0, like t_4 in the Figure 1. Tasks in this zone are the most emergent ones. They couldn't wait any longer but get the control of the processors immediately, or else their deadline is sure to be missed.

For every time unit, L-RE coordinates refresh once and all the nodes' locations are changed. A task move around the coordinates. Note that once a node appears in the coordinates, it only has two directions to move: left and down, a scale per unit time. Table 1 lists a node's actions and their meanings.

표 1. 좌표 중 점의 행위 및 그 행위의 의미
Table. 1 Nodes' actions and meanings

Action	Meaning
Appear	A new task arrives.
Move left	Task is active and is executed for a time unit.
Move down	Task waits for a time unit and isn't executed.
Move to x-axis	Task is accomplished.
Move to y-axis	Task must get the control of processor immediately.
Move below x-axis	Task will miss the deadline. Scheduling fails.

There are some other characteristics we can read from the coordinates. For instance, nodes in the line whose gradient is -1, like line 2, have the same time span to their deadlines, like t_2 and t_3 in Figure 1; the sum of a node's abscissa and ordinate decides the task's deadline: the larger the sum, the further the deadline, for example, t_2 's deadline is earlier than t_5 's.

Using the properties we mentioned above, we have found that L-RE coordinates is also suitable for describing the EDF scheduling and LLF scheduling. Furthermore, when considering the nodes's movement, it's easy to find the disadvantages of EDF algorithm and LLF algorithm. For EDF algorithm, only a new node might affect the other existent nodes' priorities. It won't change by itself as the execution goes on while no new node arrives. So even when there are only a few tasks, the task scheduling is also not feasible. For example, if there are two processors, the scheduling of $t_1(8,6)$, $t_2(8,7)$ and $t_3(10,2)$ on at any instance in time could never satisfy the deadline constraint by EDF algorithm, as shown in Figure 2(a). For LLF algorithm, if there are some nodes having a similar laxity, the processor switch would happen frequently. Also taking $t_1(8,6)$, $t_2(8,7)$, $t_3(10,2)$ and two processors for instance, the scheduling can be successfully accomplished by LLF algorithm, but one processor will have a context switch and the other one will have six context switches, as shown in Figure 2(b).

3.4 Scheduling Rules

Based on the L-RE coordinates, we propose the scheduling algorithm as follows:

- Step1: for every node, $mark = 0$; count the number of nodes in the coordinates, note it as n . If $n < m$ (m is the number of the processors in the system), allocate n processors to these n tasks randomly and mark every node as $mark = 1$; else
- Step2: if there are k node whose $l = 0$, mark them as $mark = 1$; and
- Step3: calculate the summation of abscissa and ordinate for the rest nodes, $sum = l + r$; choose $m-k$ nodes with the least sum and mark them as

$mark = 1$; especially, if there are two nodes with the same sum, the task with less laxity will be marked; among all the nodes whose $mark = 0$, find the one with the least laxity and note its laxity as q . As every time unit passes,

• $q = q - 1$;

for the nodes whose $mark = 1$, their abscissas self-minus by 1, $r = r - 1$, and ordinates l don't change ;

for the nodes whose $mark = 0$, their ordinates self-minus by 1, $l = l - 1$, and abscissas r don't change ;

this process goes on till

- Case 1: if a new node appears in the coordinates, go to Step1;
- Case 2: if a node arrives at y-axis, go to Step1;
- Case 3: if $q = 0$, go to Step1.

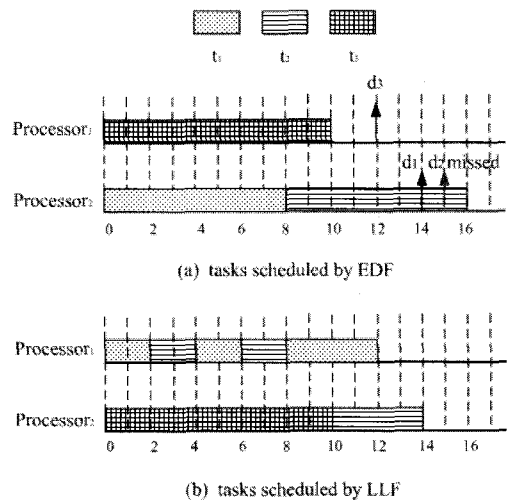


그림 2. EDF와 LLF 알고리즘의 단점
Fig. 2 Shortcoming of EDF and LLF algorithm

IV. EXAMPLE

To analyse the proposed scheduling algorithm's performance, we use an example to illustrate it. Assume there are two processors and five tasks. The tasks' parameters are listed in Table 2.

표 2. 태스크의 파라미터
Table 2. Tasks' parameters

task t_i	arrival time a_i	execution time e_i	deadline d_i
t1	0	6	11
t2	0	7	14
t3	0	9	11
t4	6	4	13
t5	9	4	16

We schedule these tasks using EDF, LLF and L-RE coordinates algorithm respectively and show the results in Figure 3.

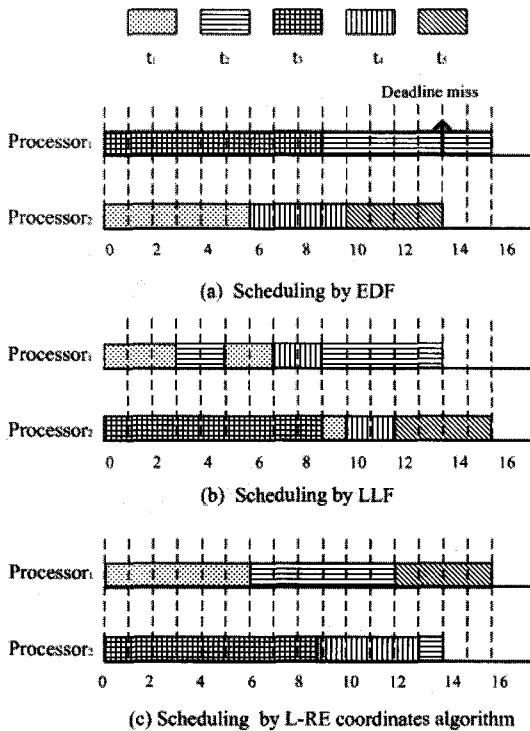


그림 3. 알고리즘의 비교
Fig. 3 Comparison of three algorithms

As showing in the figure, the scheduling is easy to be implemented by EDF algorithm, but it couldn't guarantee to schedule the given tasks in a feasible way. For example, task t_2 misses the deadline before

it finishes. LLF can successfully schedule all the tasks. However the context switch happens too frequently, total seven times. Our proposed algorithm can also accomplish all the tasks before their deadlines with four context switches. It shows that L-RE coordinates can avoid the shortcomings of both EDF and LLF.

V. CONCLUSION

This paper proposes an improved dynamic priority-driven scheduling algorithm for real-time multiprocessor system — L-RE coordinates algorithm. L-RE coordinates is a new way of improving the task scheduling efficiency. By using it, the specification of the task's property is not merely expressed as a set of numbers. The algorithm knows the task's state by illustrating the node's location and movement, and rearranges the priority to not miss the deadline. To demonstrate the algorithm's feasibility, we release a concrete scheduling problem and analyse the task scheduling sequence of three algorithms: EDF, LLF and L-RE coordinates algorithm. The analysis result indicates that our method combines the advantages of both EDF and LLF algorithm and avoids their shortcomings of poor schedulability and frequent context switches.

REFERENCE

- [1] C. L. Liu, J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment". In Journal of the Association for Computing Machinery, 1973, Pages: 46-61.
- [2] N. C. Audsley, A. Burns, M. F. Richardson, A. J. Welling, "Hard Real-Time Scheduling: The Deadline-Monotonic Approach". In proceedings of 8th IEEE Workshop on Real-Time Operating

Systems and Software, Atlanta, CA. USA, 1991, Pages: 133-137.

- [3] S. Saez, J. Vila, A. Crespo, "A Dynamic Real-Time Scheduler for Shared Memory Multiprocessors". In the 8th EUROMICRO Workshop on 'Real-Time Systems, June 12-14, 1996, Pages: 158-163.
- [4] K. Ramamritham, "Dynamic Priority Scheduling". In Real-Time System: Specification, Verification and Analysis, M. Joseph, Ed. Prentice-Hall, 1995, Pages: 66-96.
- [5] V. Salmani, M. Naghibzadeh, A. Habibi, H. Deldari, "Quantitative Comparison of Job-level Dynamic Scheduling Policies in Parallel Real-time Systems". In the proceeding of TENCON 2006. 2006 IEEE Region 10 Conference, Nov. 2006, Pages: 1-4.
- [6] J. Goossens, and P. Richard, "Overview of real-time scheduling problems". In the proceeding of the ninth international workshop on project management and scheduling, Nancy, France, April26-28 2004, Pages: 13-22.
- [7] Jean J. Labrosse, "MicroC/OS-II The Real-Time Kernel", CMP Books, 2002, Pages: 117-144.
- [8] Qing Li, Caroline Yao, "Real-Time Concepts for Embedded Systems", CMP Books, 2003, Pages: 65-78.

저자 소개



Yue Huang

Software College, Kyungwon University

〈Research interest〉 Free software, embedded system.



Yong Soo Kim

Software College, Kyungwon University

〈Research interest〉 operating system, free software, embedded system, performance management.