멀티미디어 지원을 위한 다중 프로세서 시스템에서 실시간 스케쥴링 기법

임 순 영*, 이 재 완*, 전 칠 환**

A Real-Time Scheduling Mechanism in Multiprocessor System for Supporting Multimedia

Soon-Young Rim*, Jae-Wan Lee*, Chil-Hwan Jeon**

요 약

본 논문에서는 멀티미디어 환경등의 실시간 시스템에서 타스크에게 빠른 응답시간을 제공하는 실시간 스케쥴링 기법을 제시하고자 한다.

타스크를 주기적 타스크와 비주기적 타스크로 구분하고 타스크의 형태를 긴급 타스크, 필수 타스크, 일반 타스크로 구분하여 다양한 타스크의 처리를 원하는 멀티미디어 환경에서도 적용할수 있도록 하며, 주기적 타스크의 경우 마감시간을 지키는 범위 내에서 최대한 수행을 연기하여 비주기적 타스크에 빠른 응답시간을 제공한다.

각 타스크는 우선순위를 가지며 우선순위의 변경은 동종 타스크 에서만 허용하고 동적 우선순위 방법을 이용하여 스케쥴링한다. 긴급타스크의 경우 어떠한 경우에도 수행할 수 있도록 하며 준비된 실시간 타스크가 여러 개인 경우 laxity값이 작은 타스크를 먼저 수행 할 수 있도록 한다.

본 논문에서는 비주기적 타스크가 있을 때와 없을 때를 구분하여 비주기적 타스크가 있을 때에는 주기적 타스크를 뒤로 미루고 비주기적 타스크를 앞에서 먼저 처리하며 비주기적 타스크가 없을 때에는 주기적 타스크를 앞에서 처리하는 스케쥴링 기법을 사용한다. 시뮬레이션 결과 다중 프로세서 시스템에서 지금까지 suboptimal 하다고 알려진 EDZL 보다 응답시간 관점에서 제안한 기법이 우수하였고 부하가 증가 하여도 성능이 우수하였다.

Abstract

This paper presents a mechanism which supplies tasks with fast turn-around time on real-time multimedia environments.

* 군산대학교 정보통신공학과

** 군산대학교 전기공학과

접수일자 : 1997년 9월 12일

Tasks are classified into periodic and aperiodic tasks according to their executing period, and the types of them are classified into three groups : critical tasks, essential tasks and common tasks by the degree of its urgency. In the case of periodic tasks, we defer the execution of it within the extent to keep the deadline as long as possible and serve the aperiodic tasks, and provide aperiodic tasks with fast turn-around time.

Changing the priority of each task is allowed within the same type and it is scheduled by using the dynamic priority. The emergency tasks are executed within deadline in any circumstances, and the least laxity one is served first when many real-time tasks are waiting for execution.

The result of simulation shows that the proposed mechanism is better than the EDZL, known as suboptimal in multiprocessor systems, in the point of turn-around time.

Ⅰ. 서 론

컴퓨터 성능의 향상과 네트워크 기술의 발전 및 데이터의 멀티미디어화에 따라 컴퓨터 환경은 많은 변화를 가져오고 있다. 이러한 시스템은 실시간 처리를 필요로 하며 이에 따라 타스크들의 실시간성 및 주기성을 고려한 실시간 스케쥴링 기법들이 많이 연구되고 있다.

최근 많이 보급되고 있는 멀티미디어 컴퓨터는 데이터 공유에 대한 요구가 급증하고 있으며 또한 많은 데이터를 가지고 있는데 특히 텍스트(Text), 이미지(Image), 오디오(Audio), 비디오(Video)등의 실시간 처리를 요하는 여러 형태의 데이터를 가지고 있고 이러한 타스크들은 주기적 타스크와 비주기적 타스크를 포함하고 있다. 경성 마감시간(hard deadline)을 가지는 주기적 타스크와 연성 마감시간(soft deadline)을 가지는 비주기적 타스크를 함께 스케쥴링할 필요가 있을 때 실시간 시스템은 일반적인 시스템과는 달리 수행의 정확성뿐만이 아니라 일정한 마감시간 내에 타스크를 수행할 수 있는지가 중요한 척도가 된다[1,5].

수행하는 과제들이 단순하다면 사건들을 하나의 컴퓨터에서 수행하는 것이 가능하지만 대부분의 실시간 시스템들은 복잡해서 여러 개의 activity들을 조정 통제하기 위해서는 여러 개의 프로세서 즉 다중 프로세서와 복잡한 s/w 구조를 필요로 하고 이러한 복잡성의 조정과 멀티미디어 데이터 처리를 위해 다중 프로세서 상에서 실시간 스케쥴링 기법이 필요하다[2].

이에 따라 실시간 스케쥴링 알고리즘에 대한 연구들이 국내외로 수행되고 있으며 또한 연구방향도 단일 프로세서 상에서 점차적으로 다중 프로세서 및 분산 시스템으로 발전하고 있고[6,7] 주기적 타스크와 비주기적 타스크를 함께 스케쥴링할 때 마감시간 내에 정확하게 끝내는 것도 중요하지만 비주기적 타스크에 빠른 응답시간을 제공하는 것도 비주기적 타스크 스케쥴링의 목표이다.

본 논문에서는 멀티미디어 지원을 위한 다중 프로세서 시스템에서 타스크를 스케쥴링할 때 비주기적 타스크가 있을 때와 없을 때를 구분하여 비주기적 타스크가 있을 때에는 마감시간을 만족시키는 범위내에서 주기적 타스크를 뒤로 미루고 비주기적 타스크를 앞에서 먼저 처리하며 비주기적 타스크가 없을 때에는 주기적 타스크를 앞에서 처리하는 스케쥴링 기법을 사용함으로써 비주기적 타스크에 빠른 응답시간을 제공한다. 타스크는 수행의 긴급성에 따라 타스크의 특성을 긴급, 필수, 일반 타스크로 구분하여 긴급 타스크와 필수 타스크에 우선권을 주어 처리 한다. 각 타스크는 우선순위를 가지며 우선순위의 변경은 동종 타스크에서만 허용하고 동적 우선 순위 방법을 이용하여 스케쥴링한다.

본 논문의 구성은 2장에서 관련연구로 마감시간 우선 순위에 따라 스케쥴링 되는 EDA와 laxity값을 기준으로 스케쥴링되는 LLA, 다중 프로세서 시스템에서는 suboptimal 하다고 알려진 EDZL에 대해서 살펴보고, 3장에서는 시스템 모델 및 타스크 모델에 대해서 설명하고, 4장에서는 본 논문에서

제안하는 실시간 스케줄링 기법으로 스케줄러 구조, 주기적 타스크의 수행테이블, 비주기적 타스크의 수행테이블에 대해서 설명하고, 5장에서는 기존의 기법과 제시한 기법을 시뮬레이션을 통해 성능을 평가하고, 6장에서는 결론 및 향후 연구 방향에 대해서 기술한다.

Ⅱ. 관련 연구

타스크 스케줄링 알고리즘으로는 마감시간우선(Earliest Deadline First)알고리즘, 비율단조(Rate Monotonic)알고리즘[4], 슬랙을 이용하여 비주기적 타스크를 서비스하는 슬랙 스틸링(Slack Stealing) 알고리즘, 그리고 순환수행(Cyclic Executive)스케줄링 등이 있다[8,9,10].

슬랙 스틸링 알고리즘은 모든 종류의 비주기적 타스크에 대해서 최소 응답시간을 제공 하지만 오프라인과 온라인시 유지해야 하는 자료구조와 비주기적 타스크에 할당할 슬랙시간을 계산하는데 큰 오버헤드를 가지는 단점이 있다[3].

순환수행 알고리즘은 사전에 계산된 리스트들을 반복적으로 수행하는 구조를 갖고 경성 마감시간을 갖는 시스템에서 가장 많이 사용하는 기법이지만 이를 위해서는 프로그램내에 시간과 동기화 기법들이 삽입 되어야 하고 적절한 스케줄링을 구성하기가 힘들고 시스템의 변경으로 인해 새로운 스케줄링을 해야할 필요가 발생할 수도 있다[8].

다중 프로세서 상에서 대표적인 알고리즘으로는 EDA, LLA, EDZL등을 들수 있다[1,2].

1. EDA (Earliest Deadline Algorithm)

EDA는 마감시간을 기초로 타스크들의 스케줄링이 이루어지는 방법으로 타스크 중에서 마감시간이 짧은 타스크를 선정하여 처리하므로 문맥교환이 적게 일어나 단일 프로세서 상에서는 최적으로 알려져 있지만 다중 프로세서 상에서는 스케줄링 성능이 떨어진다[12]. n개의 주기적 타스크 집합이 아래식을 만족하면 마감시간 우선 알고리즘에서 스케줄링이 가능하다[11].

$$\frac{C_1}{T_1}+\frac{C_2}{T_2}+ \cdots\cdots\cdots +\frac{C_n}{T_n} \le U(N)=1$$

C_1 ..... C_n : 타스크 수행시간

T_1 ..... T_n : 타스크 주기

U(N) : 프로세서 이용률

위 식에서와 같이 프로세서 이용률이 1이하이면 타스크들이 필요로 하는 시간의 총량이 프로세서 시간보다 작아지므로 마감시간을 넘기지 않는 스케줄링이 가능하다.

2. LLA (Least Laxity Algorithm)

LLA는 계속 동적으로 변하는 laxity값에 의하여 스케줄링 되므로 문맥교환 오버헤드가 크다는 단점이 있지만 다중 프로세서 시스템에서 좋은 스케줄링 성능을 갖고 있다고 알려져 있다. EDA와 LLA의 스케줄링 성능을 비교해 보면 다음과 같다.

(2,3), (3,5), (5,6)이라는 준비 타스크를 2개의 프로세서에 할당할 때 EDA는 마감시간 우선순위에 따라 (2,3), (3,5)타스크를 할당하고 (5,6)타스크를 할당하므로 마감시간을 지키지 못해 스케줄링이 실패하지만 LLA는 laxity값이 작은 타스크를 laxity 값이 같을 때에는 마감시간에 가까운 타스크를 할당하므로 (2,3), (5,6)타스크를 할당하고 (3,5)타스크를 할당하므로 스케줄링에 성공함을 알 수 있다. 다시말하면, EDA는 다중 프로세서 환경에서는 스케줄링 성능이 현저히 떨어짐을 의미하고 LLA는 높은 문맥교환 오버헤드를 제외하면 뛰어난 스케줄링 성능을 지니고 있어 다중프로세서 시스템에서 Suboptimal한 특성을 가지고 있다[12].

또한 다중 프로세서 시스템에서는 마감시간이나 laxity중 어느 하나 만으로는 효과적인 스케줄링을 할 수 없다고 알려져 있고 이러한 문제점을 해결하기 위해 마감시간과 laxity 두 가지 요소를 사용하는 알고리즘이 EDZL이다.

3. EDZL (Earliest Deadline Zero Laxity Algorithm)

EDZL은 기본적으로 EDA에 기초하고 있고 위급한 상황이 발생시 laxity값에 의해 스케줄링 한

다. 즉 큐(Queue)안에서 대기하는 타스크들 중에 laxity값이 0인 타스크가 발생했을 때는 먼저 스케줄링 되어진다. EDZL이 다중 프로세서 시스템에서는 LLA가 스케줄링하는 모든 준비 타스크들의 집합을 LLA와 마찬가지로 스케줄링할 수 있고 문맥교환 오버헤드는 EDA와 거의 같으면서 스케줄링 성능은 LLA와 같아 다중 프로세서 시스템에서는 suboptimal한 알고리즘으로 알려져 있다[12].

Ⅲ. 시스템 및 타스크 모델

1. 시스템 모델

시스템의 스케줄링 단위는 타스크이며 순환수행 스케줄링 기법의 경우와 같이 스케줄링에 필요한 정보는 코드 속에 내포 된다고 가정한다. 타스크가 도착할 때 다음 사항은 주어진다고 가정하고 비주기적 타스크의 경우 본 논문에서 제안한 기법에 따라 처리한다.

- o 주기적 또는 비주기적
- o 선점 또는 비선점
- o 타스크의 특성
- o 최악의 경우 수행시간
- o 마감시간
- o 필요한 자원

2. 타스크 모델

타스크 T는 (C, D)로 구성되어 있고 C는 타스크 T를 완료 하는데 필요한 수행시간이고 D는 타스크 T가 완료 되어야 할 시간으로 마감시간을 나타낸다. 타스크 T에 대한 수행시간 C와 마감시간 D는 타스크 T가 수행됨에 따라 변화한다. 타스크 T의 마감시간까지 남아 있는 시간을 C(i), 마감시간을 D(i)라고 한다면 시간 I 에서 타스크의 laxity는 Laxity(i) = D(i) - C(i)로 정의 된다.

laxity는 긴급한 타스크의 척도이고 laxity가 0인 타스크는 즉시 실행 되어야 하며 Interrupt되어서는 않되고 또한 laxity값이 음수 일때는 마감시간을 지키지 못할 수도 있다는 것을 나타낸다. 시간(i)에서 요구하는 n개의 프로세서와 m개의 타스크들이 있다고 가정하면(m>n) 어떤 시간에서 m개의 타스크중 n개를 실행할 수 있고 시간 I 에서 실행되고 있는 토큰에 대한 새로운 위치는 L(i+1) = L(i), C(i+1) = C(i) - 1 이고 시간 I+1에서 실행되고 있지 않는 토큰에 대한 위치는 L(i+1) = L(i) - 1, C(i+1) = C(i)로 주어진다[14]. 타스크의 종류는 주기적 타스크와 비주기적 타스크로 구분하며 타스크는 특성에 따라 긴급타스크, 필수타스크, 일반타스크로 분류하여 타스크는 각 타스크 내에서 우선순위를 갖고 있으며 긴급 타스크는 어떤 경우에도 우선적으로 실행 되어야 한다.

긴급타스크는 어떤 환경에서도 마감시간을 준수하여야 하는 타스크이며 이를 지키지 못하였을 경우에는 심각한 문제가 발생한다. 긴급 타스크는 최악의 경우에도 마감시간을 지키게 함으로써 경성 마감시간(hard deadline)을 갖는 스케줄링이 된다. 필수타스크는 마감시간을 가지면서 시스템운영에 중요한 타스크로서 마감시간을 지키지 못하였을 경우 상당한 시스템의 성능저하를 가져올 수 있으며 수행을 위해 요구되는 충분한 자원을 확보하지 못했을 경우에는 긴급타스크에 비해 융통성을 가지며 연성 마감시간(soft deadline)스케줄링 기법이 적용된다. 실시간 타스크는 주기적 타스크와 비주기적 타스크중에서 긴급타스크와 필수 타스크로 구성된다.

일반 타스크는 비실시간 타스크로서 일상 응용 타스크이고 시스템 고장이나 부하가 많을때에는 쉽게 선점된다.

Ⅳ. 실시간 스케줄링 기법

다중 프로세서 시스템에서 suboptimal하다고 알려진 EDZL의 경우 마감시간과 laxity값만을 가지고 타스크를 할당 하므로 성공률 면에서는 우수할지 모르지만 실시간 시스템과 비실시간 시스템이 혼합된 환경에서의 비실시간 타스크의 응답시간 면에서는 다루어지지 않고 있기 때문에 본 논문에서는 이러한 환경에서 빠른 응답시간을 제공하는 스케줄링 기법을 제시하고자 한다.

타스크의 종류를 주기적 타스크와 비주기적 타스크로 구분하였고 주기적 타스크는 마감시간을 만족시키는 범위내에서 수행을 연기 시킴으로서 비주기적 타스크에 빠른 응답 시간을 제공하고[9],

비주기적 타스크는 타스크의 형태를 긴급타스크, 필수타스크, 일반타스크로 구분하여 멀티 미디어 환경에서도 다양한 타스크를 처리할 수 있도록 하였으며 비주기적 타스크는 타스크의 형태에 따라 순환수행 기법을 기반으로 한 동적 우선순위 방법을 이용하여 스케쥴링 한다[13].

다중 프로세서 시스템에서 마감시간을 갖는 모든 실시간 타스크들을 스케쥴링 하는 최적화(Optimal)된 알고리즘은 존재하지 않으며 준비(Ready)상태에 있는 타스크를 대상으로 하는 suboptimal한 온라인 스케쥴링 알고리즘 연구가 주요 대상이 된다. 다중 프로세서 시스템에서 스케쥴링 문제에 대한 해결은 그림 1과 같이 타이밍 다이어그램과(Timming Diagram)과 간트챠트(Gantt Chart)로 나타낼 수 있다[5].

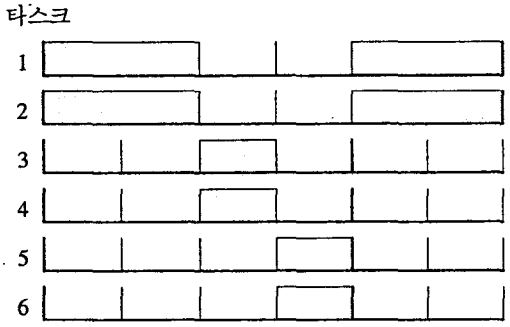


그림 1. 타이밍 다이어그램과 간트챠트
Fig. 1. Timming Diagram and Gantt Chart

1. 스케쥴러 구조

본 논문에서 제안하는 스케쥴러 구조는 그림 2와 같다.

주기적 타스크는 미리 정적으로 계산된 주기정보에 따라 하이퍼 주기를 계산하며 각 타스크들의 수행시간은 마감시간을 만족하는 범위 내에서 최

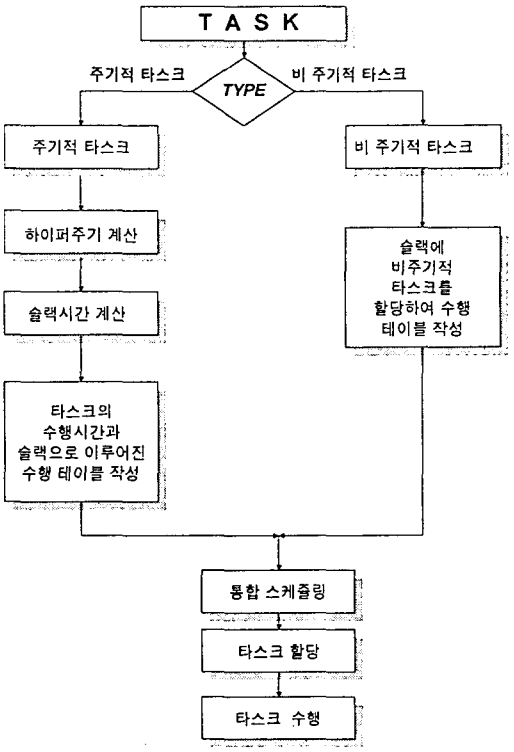


그림 2. 스케쥴러 구조
Fig. 2. The structure of scheduler

대한 연기시킴으로서 슬랙시간(slack time)을 가능한 앞에 두고 이 슬랙 시간에 비주기적 타스크를 할당한다.

비주기적 타스크가 있을 때와 없을 때를 고려한 타스크의 수행 테이블의 예는 그림 3과 같다.

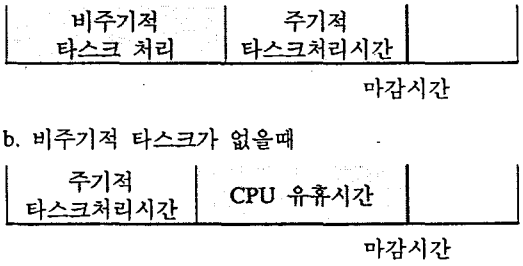


그림 3. 타스크의 수행 테이블
Fig. 3. Processing table for tasks

위와 같은 스케줄링 기법을 사용함으로써 다중 프로세서 시스템에서 suboptimal하다고 알려진 EDZL 보다 비주기적 타스크에 빠른 응답시간을 제공할 수 있다는 장점이 있다.

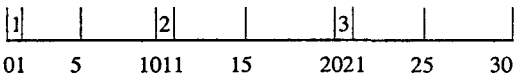
2. 수행 테이블 작성

2.1 주기적 타스크의 수행 테이블

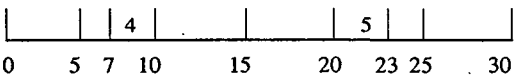
주기적 타스크는 마감시간 내로 최대한 수행을 연기하고 비주기적 타스크는 본 논문에서 제안한 기법에 따라 긴급 타스크와 필수 타스크에 우선권을 주며 실시간 타스크는 우선 순위에 따라 타스크를 처리하고 우선순위의 변동은 동종 타스크 내에서만 가능하다. 즉 일반 타스크는 우선 순위가 높아져도 필수타스크로 바뀔 수 없고 필수타스크는 우선 순위가 아무리 높아져도 긴급타스크로 바뀔 수 없다. 주기적 타스크 τ i(Ci, Ti, Di) 로 나타낼 수 있는데 주기적 타스크의 준비 타스크가 τ_1 (1, 10, 10), τ_2 (3, 15, 15), τ_3 (2, 10, 10), τ_4(2, 15, 15) 라고 할 때 주기적 타스크의 수행 테이블은 그림 4와 같고 작성된 수행 테이블에 따라 다중 처리기에 타스크를 할당한다.

■ 준비 타스크 (주기적 타스크)

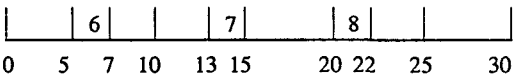
τ_1 (1,10,10)



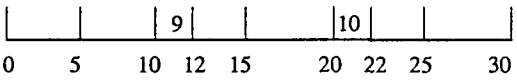
τ_2 (3,15,15)



τ_3 (2,10,10)

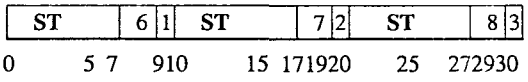


τ_4 (2,15,15)

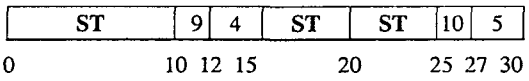


■ 프로세서에 할당한 경우

Processor 1



Processor 2



(ST : Slack time 으로 비주기적 타스크 처리시간)

그림 4. 주기적 타스크의 수행 테이블
Fig. 4. Scheduling table for periodic tasks

그림 4에서 4개의 주기적 타스크 τ_1(1, 10, 10), τ_2(3, 15, 15), τ_3(2, 10, 10), τ_4(2, 15, 15)에 대해 작성된 슬랙시간(ST: Slack Time)은 비주기적 타스크를 처리할 수 있는 시간으로 주기적 타스크를 처리하고 남은 시간이며 마감시간을 넘기지 않는 범위 내에서 주기적 타스크를 뒤로 연장하여 얻을 수 있는 시간이다.

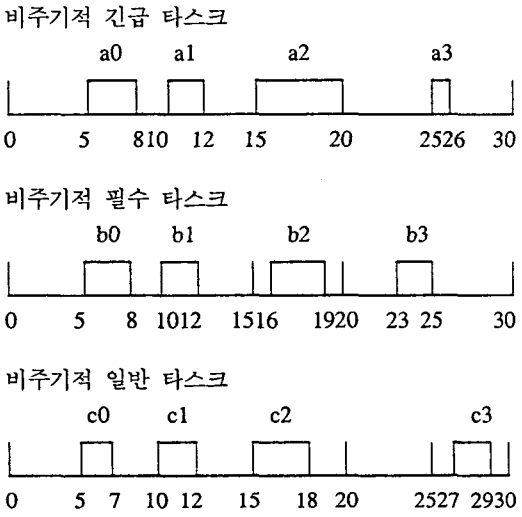
여기에서 하이퍼 주기는 각 주기적 타스크의 최소 공배수 이다. 하이퍼 주기마다 주기적 타스크들은 같은 형태의 도착형태를 반복하게 된다. 그림 4는 주기적 타스크의 주기가 τ_1=10, τ_2=15, τ_3=10, τ_4=15이므로 하이퍼 주기는 30이다. 그림 4에서 나타난 바와 같이 주기적 타스크를 스케줄링할 경우 비주기적 타스크를 실행할 수 있는 시간을 가능한 각 주기의 앞에 확보 하였다.

2.2 비주기적 타스크의 수행테이블 작성

비주기적 타스크에 대한 수행 테이블은 주기적 타스크의 수행 테이블 상에서 작성되며 비주기적 타스크는 긴급타스크, 필수타스크, 일반타스크 모두를 포함하고 비주기적 타스크의 우선 순위가 이미 작성된 주기적 타스크 수행테이블의 우선순위보다 높을 때에는 우선 순위에 따라 수행 테이블을 수정하여 완성하게 되며 전체 타스크 할당 시간보다 슬랙시간이 작을 경우에 남아있는 타스크는 오버로드(overload)로 간주한다. 비주기적 실시간 타스크가 많을 때에는 우선 순위에 따라 수행 테이블을 작성한다. 비주기적 타스크가 없을 때에

는 그림 3 의 b 처럼 뒤로 미루어 놓았던 주기적 타스크를 앞에서 처리 한다.

■ 준비 타스크 (비주기적 타스크)



비주기적 준비 타스크가 위와 같이 주어져 있을 때 주기적 타스크상에서 만들어진 수행 테이블에 준비된 비주기적 타스크(긴급, 필수, 일반)를 할당한 비주기적 타스크의 수행 테이블은 그림 5와 같다.

■ 프로세서에 할당한 경우

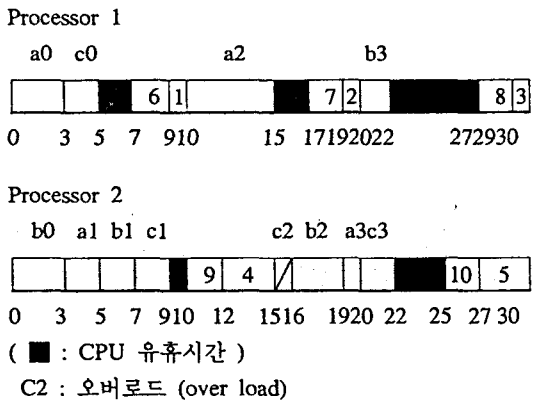


그림 5. 비주기적 타스크의 수행 테이블
Fig. 5. Scheduling table for aperiodic tasks

타이밍 다이어그램과 간트챠트 방법에의해 다중 프로세서에 준비 타스크를 할당할 때 C2 타스크의 경우 수행 하는데 필요한 수행 시간은 3인데 프로세서상에 슬랙시간은 1이기 때문에 1만큼만 수행하고 준비 타스크의 남은 수행시간 2는 오버로드로 간주한다.

V. 성능평가

1. 시뮬레이션 환경

응답시간은 마감시간에서 도착시간을 뺀 시간이고 평균 응답시간은 전체 응답시간을 전체 부하로 나눈 것이다. 비주기적 타스크의 도착시간 간격은 평균이 20인 지수분포로 고정 시키고 비주기적 타스크의 도착시간 간격을 변화 시키면서 즉, 시스템 부하를 변경시켜 부하에 따른 기존의 기법과 본 논문에서 제안한 기법간의 성능을 분석 한다. 비주기적 타스크의 경우 긴급 타스크는 수행시간 평균이 5인 지수분포, 필수 타스크는 수행시간 평균이 7인 지수분포, 일반 타스크는 수행시간 평균이 10인 지수분포를 사용 하였고 시뮬레이션은 개인용 컴퓨터(pentium pro)와 C 언어를 사용 하였다.

2. 성능평가

실시간 타스크와 일반타스크가 함께 수행되는 환경에서 기존의 기법과 본 논문에서 제안하는 비주기적 타스크가 있을 때와 없을 때를 구분하여 비주기적 타스크가 있을 때에는 주기적 타스크를 뒤로 미루고 비주기적 타스크를 앞에서 먼저 처리하고 비주기적 타스크가 없을 때에는 미루어 놓았던 주기적 타스크를 앞에서 처리하는 기법을 시뮬레이션을 통해 성능을 비교 평가하였다. 시뮬레이션 결과 그림 6에 나타난 바와 같이 기존의 기법(△△△)의 평균치(일반타스크, 필수타스크, 긴급타스크) 보다 제안한기법(○○○)의 평균 응답시간이 약 25%정도 감소함을 알 수 있었고 또한 부하가 증가 하여도 성능이 안정됨을 알 수 있었다. 시뮬레이션 결과에 대한 데이터값은 부록으로 첨부하였다.

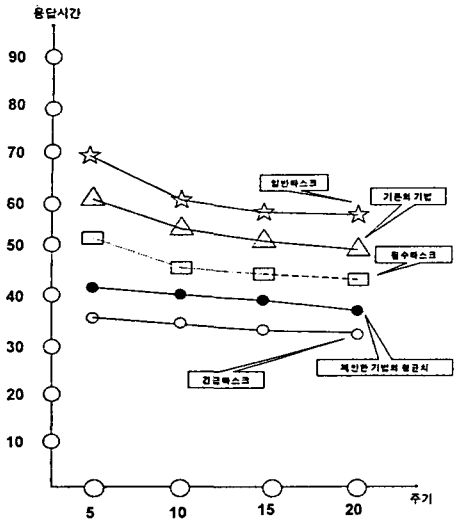


그림 6. 평균응답시간
Fig. 6. The comparison of average response time

VI. 결론 및 향후 연구방향

본 논문은 다중 프로세서 시스템에서 주기적 타스크와 비주기적 타스크가 혼합된 멀티미디어 환경에서 실시간 타스크들의 마감시간내 수행을 보장 하면서 비주기적 타스크에 빠른 응답시간을 제공하는 실시간 스케줄링 기법을 제시 하였다. 비주기적 타스크가 있을 때와 없을 때를 구분하여 비주기적 타스크가 있을 때에는 주기적 타스크를 뒤로 미루고 비주기적 타스크를 앞에서 먼저 처리하고 비주기적 타스크가 없을 때에는 미루어 놓았던 주기적 타스크를 앞에서 처리 하도록 하였다. 비주기적 타스크는 긴급, 필수, 일반 타스크로 구분하여 스케줄링함으로써 긴급 타스크에 대해서는 반드시 마감시간을 지키는 경성 마감시간 방법을 적용하였고, 필수 타스크에 대해서는 연성 마감시간 방법을 적용하였다. 기존의 방법과 제안한 방법을 시뮬레이션을 통해 성능을 평가한 결과 일반 타스크의 평균 응답시간이 25%정도 감소함을 알 수 있었고, 부하가 증가 하여도 성능이 안정됨을 알 수 있었다. 앞으로 분산 처리 환경과 멀티 미디어 서버 시스템에도 적용할 수 있도록 계속적인 연구가 필요하다.

참고문헌

[1] J. P. Lehoczky and S. Ramos-Thuel, " An Optimal Algorithm for Schedling Soft-Aperiodic Task in Fixed-Priority Preemptive Systems", Proceeding of the IEEE Real-Time System Symposium, Dec. 1992, pp.110-123.

[2] L. Sha, B.Sprunt and J. P. Lehoczkey, "Aperiodic Task Scheduling for Hard Real-Time System", The Jurnal of Real-Time System 1989, pp.27-69.

[3] R. I. Dvis, K. W. Tndel, and a. Bms, "Sheduling Sack Time in Fixed-Priority Preemptive Systems", Proceeding of the IEEE Real-Time System Symposium, Dec. 1993, pp.222-231.

[4] J. P. Lehoczky, 1. Sha, and Y. Ding, "The Rate Monotonic Scheduling Algorithm : Exact Characterization and Average Case Behavior", Technical Report Department of Statics, Carnegie Mellon University, Pittsburgh, PA, 1987.

[5] John A. Stankovic, "The Integration of Scheduling and Fault Tolerance in Real-Time Systems", Proceeding of COINS Technical Report 92-49, Mar. 1992.

[6] Babak Hamidzadeh and David J. Lilja, "Dynamic Scheduling Strategies for Shared-Memory Multiprocessors", Proceeding of the IEEE 16th ICDCS 1996, pp.208-215.

[7] Rajiv Gupta, Daniel Mosse and Richard Suchoza, "Real-Time Scheduling Using Compact Task Graphs", Proceeding of the IEEE 16th ICDCS 1996, pp.55-62.

[8] C. D. Locke. "Software Architecture for Hard Real-Time Applications: Cyclic Executives vs. Fixed Priority Executives", The Journal of Real-Time Systems, 4(1):37-53, Mar. 1992.

[9] 임순영, 이경복, 윤인숙, 이재완, "멀티미디어 지원을 위한 실시간타스크 스케줄링 기법",

한국 정보 처리학회 봄 학술 발표 논문집, 제 22권 1호, 1997, pp.886-890.

[10] 김태웅, 신현식, "고정 우선순위 실시간 시스템에서 비주기 테스크 스케줄링 알고리즘", 95년도 한국 정보 과학회 봄 학술 발표 논문집, 제 22권 1호, 1995, pp.367-369.

[11] C. L. Liu, J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", Journal of the ACM, Vol.20, No1, Jan. 1973, pp.46-61.

[12] 이석균, 임준택, "실시간 온라인 스케줄링 알고리즘들의 분석 및 성능비교", '95한국 정보 과학회 봄 학술 발표 논문집, Vol.22, No.1, 1995, pp.383-386.

[13] 임순영, 이경복, 윤인숙, 이재완, " 다중 프로세서 시스템에서 실시간 스케줄링 기법", 97 전북지역 전자, 통신 하계합동 학술대회 논문집, 제6권 1호, 1997, pp. 91-95.

[14] Michael L. and Mek. Dertouzos, Aloysius. : "Multiprocessor On-Line Scheduling of Hard-Real-Time Tasks". IEEE Trans. on software Engineering, Vol.15, No.12 Dec. 1989, pp. 1497-1506.

부 록

1. 시뮬레이션 결과 데이타값

도착시간 간 격	응 답 시 간		응 답 시간 감소율(%)
	기존의 기법 평 균 치	제안한 기 법	
5	60	41	32.27(68.3)
10	52	40	23.08(76.9)
15	50	39	22 (78)
20	48	37	22.92 (77)

임 순 영(Soon-Young Rim)

1992년 2월 전북산업대학교 전자계산학과 졸업(이학사)

1995년 8월 수원대학교 교육대학원 전자계산학과 졸업(전산 교육학 석사)

1998년 2월 군산대학교 대학원 정보통신공학과 졸업(공학석사)

1984년 10월~1994년 11월 제일군산 컴퓨터 통신학원 원장겸 강사

1993년 12월~현재 동광통신 주식회사 기술이사

1997년 3월~현재 군장공업전문대학교 전자계산학과 겸임교수

관심분야 : 운영체제, 분산시스템, 데이터베이스, 멀티미디어, 전자도서관

이 재 완(Jae-Wan Lee)

1984년 2월 중앙대학교 전자계산학과 졸업(이학사)

1987년 8월 중앙대학교 대학원 전자계산학과 졸업(이학석사)

1992년 2월 중앙대학교 대학원 전자계산학과 졸업(이학박사)

1987년 11월~1992년 8월 한국체육과학연구원 선임연구원

1996년 3월~1997년 1월 한국학술진흥재단 전문위원(파견)

1997년 1월~1998년 2월 한국학술진흥재단 부설 첨단학술정보센타 전문위원(겸직)

1992년 9월~현재 군산대학교 정보통신공학과 조교수

관심분야 : 운영체제, 분산시스템, 데이터베이스, 멀티미디어, 전자도서관

전 칠 환(Chil-Hwan Jeon)
1980년 10월 서울대학교 전기공학과 졸업(공학사)
1986년 5월 Vanderbilt University 대학원 졸업(전기공학 석사)
1990년 5월 Vanderbilt University 대학원 졸업(전기공학 박사)
1980년 9월～1983년 4월 한국기계연구소 연구원
1996년 3월～1997년 1월 군산대학교 공학연구소 소장
1990년 9월～현재 군산대학교 공대 전기공학과 부교수
1995년～현재 한국전기학회 전북지회 이사
1996년 10월～현재 전라북도 건설심의위원
1997년 2월～현재 군산대학교 공대 학장
관심분야 : 전력전자, 제어공학, 전기기계