Journal of Digital Contents Society Vol. 9 No. 3 Sept. 2008(pp. 441-448)

실시간 비주기 태스크 스케줄링을 위한 개선된 합성 이용율에 관한 연구

문석환*, 김인국**

요 약

최근 비주기 태스크들의 스케줄링 분석을 위한 많은 알고리즘이 제시되었는데, 그중 임의의 시점에 비주기 태스크들의 스케줄링 가능성을 판단하기위한 알고리즘으로서 합성 이용율(synthetic utilization)이 Abdelzaher등에 의해 제시되었는데, 이들은 임의의 시점에 합성이용율의 상한 값인 $\frac{1}{1+\sqrt{1/2}} \approx 0.59$를 넘지 않으면 비주기 태스크들이 스케줄링 가능 하다는 것을 증명 하였다. 하지만 이 방법은 비주기 태스크들의 프로세서 이용율 계산 시 태스크가 실제 모든 실행시간을 종료하여 더 이상의 실행시간을 갖지 않더라도 현재요청집합(current invocation)에 속해 있다면 실행시간과 종료시한을 합성 이용율에 포함하기 때문에 실제 스케줄링 가능한 태스크들이 실행 불가능한 경우로 판단되는 문제점을 가지고 있다. 본 논문에서는 이러한 문제점을 해결하여 더 많은 비주기 태스크들이 스케줄링 가능 하도록 개선된 합성 이용율 방법을 제시하였다.

A Study on Improved Synthetic Utilization for Real-Time Aperiodic Tasks Scheduling

Seok-Hwan Moon*, In-Guk Kim**

Abstract

Recently, several algorithms for scheduling aperiodic tasks have been proposed. Among them, Abdelzaher et al proposed an algorithm to determine the schedulability of aperiodic tasks, and proved that the aperiodic tasks are schedulable if the upperbound of synthetic utilization is less than or equal to $\frac{1}{1+\sqrt{1/2}} \approx 0.59$. But this algorithm has a drawback in that if some tasks, even though they are completed and have no more execution times, are included in the current invocation set, their execution times and deadlines are added to the synthetic utilization. This may lead to a problem in which actually schedulable tasks are decided not to be schedulable. In this paper, we recognize the above mentioned problem and propose an improved synthetic utilization method that can be used to schedule aperiodic tasks more efficiently.

Keywords : Real-Time Scheduling, Aperiodic Tasks, Synthetic Utilization

1. 서론

실시간 시스템(real-time system)은 태스크의 수행결과의 정확성뿐만 아니라 처리시한의 제한인 종료시한(deadline)을 엄격하게 지키는 것이 요구되는 시스템을 말한다. 이러한 실시간 시스템은 종료시한이 엄격하게 지켜져야 하는 경성 실시간(hard real-time) 시스템과 그렇지 않은 연성 실시간(soft real-time) 시스템으로 구분 할 수 있다. 실시간 시스템에서 실행되는 태스크들은 일정 시간 마다 도착하여 반복적으로 실행되는 주기(periodic) 태스크와 도착 시간이 정해져 있지 않은 비주기(aperiodic) 태스크로 구분 할 수 있다. 본 논문에서는 합성 이용율을 기반으로

※ 제일저자(First Author) : 문석환
접수일자:2008년06월26일, 심사완료:2008년09월03일
* 영동대학교 임베디드 소프트웨어학과
shmoon@youngdong.ac.kr
** 단국대학교 컴퓨터학부

하여 경성 실시간 비주기 태스크들을 스케줄링 하는 기법을 다루고 있다.

이제까지 제안된 스케줄링 알고리즘들 중에서 Rate Monotonic(RM)[1] 스케줄링 알고리즘은 고정 우선순위 기반 알고리즘들 중 최적이고, Earliest Deadline First(EDF)[1][2] 스케줄링 알고리즘은 동적 우선순위 알고리즘들 중 최적인데 이들은 모두 주기적 태스크 모델을 기본으로 하고 있다. RM과 EDF는 프로세서 이용율을 이용하여 태스크들을 스케줄링 하는 방법이며, [11]에서 제시한 방법은 최악의 경우 응답시간(worst case response time)[12]를 이용하여 주기 태스크들의 스케줄링 가능성 여부를 판단하는 방법이다. 그러나 최근에는 클라이언트/서버 컴퓨팅과 같은 네트워크 환경에서 빈번하게 발생되는 비주기 태스크 처리를 위한 많은 연구가 진행되고 있다.

비주기 태스크에 대한 스케줄링 분석이 주기 태스크에 비하여 상대적으로 어려운 이유는 태스크가 도착하기 이전에는 태스크의 도착 시점, 수행 시간, 종료 시한을 주기 태스크처럼 미리 예측 할 수 없기 때문이다. 비주기 태스크를 스케줄링 하는 방법으로는 연성 실시간 비주기 태스크 스케줄링 방법과 경성 실시간 비주기 태스크 스케줄링 방법으로 구분 할 수 있는데, 연성 실시간 비주기 태스크 스케줄링은 주로 비주기 태스크의 응답 시간을 줄여 성능을 향상 시키는 방법을 사용 하고 있으며, 대표적인 방법으로 디퍼러블 서버(deferrable server, DS 서버)[3], 우선순위 교환 서버(priority exchange server, PE 서버)[3][4], 스포래딕 서버(sporadic server SS 서버)[5]등과 같은 비주기 서버 알고리즘이 제안되었다. 하지만 이러한 방법은 비주기 태스크의 종료시한을 보장하는 것보다 응답시간을 줄여 향상 시키는 방법을 사용하기 때문에 경성 비주기 태스크에 적용하기 힘들다.

반면에 경성 실시간 비주기 태스크 스케줄링 방법으로는 연성 실시간 비주기 태스크에도 적용 가능한 슬랙(slack) 시간 분석을 통한 슬랙 스틸링(slack stealing) 알고리즘[6][7]과 최근에 Abdelzaher등에 의해 제시 된 비주기 태스크들의 이용율 분석을 통한 합성 이용율 알고리즘[8][13][14][15]등이 있다.

본 논문의 구성은 다음과 같다. 2장에서는 합성 이용율과 이를 이용한 스케줄링 방법을 기술한 후 이의 문제점과 개선된 방법을 제시한다. 3장에서는 모의 실험결과를 보이고, 마지막으로 4장에서는 결론 및 향후 연구 과제에 대하여 기술한다.

2. 비주기 태스크 이용율

2.1 합성 이용율

Abdelzaher등에 의해 제시된 합성 이용률 방법[8]은 경성 비주기 태스크 집합에 대하여 스케줄링 가능한 합성 이용률의 상한 값을 제시하였다. 또한 이를 이용하여 비주기 태스크가 도착했을 때 합성 이용률을 상한 값과 비교하여 도착한 비주기 태스크의 수락여부를 인정할 것인지, 거절할 것인지를 결정하게 된다.

비주기 태스크 집합 T_a를 $\{T_1, T_2, T_3 \cdots T_i \cdots\}$라 할 때, T_1은 T_2보다 먼저 도착(arrival)된 태스크라 가정 한다. 즉 T_i는 T_{i+1}보다 먼저 도착한 비주기 태스크 이다. 비주기 태스크 T_i의 수행시간(execution time)을 $C_i(>0)$, 도착시간(arrival time)을 A_i, 상대적 종료시간(relative deadline)을 $D_i(>0)$ (여기서 절대적 종료시간(absolute deadline) d_i는 $A_i + D_i$로서 정의된다.)라 할 때 임의의 시점 t에서의 합성 이용률은 다음과 같이 정의된다.

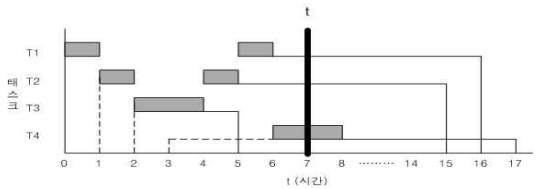
$$U_{(t)} = \sum_{T_i \in S_{(t)}} \frac{C_i}{D_i} \qquad (1)$$

식(1)은 Liu와 Layland가 정의한 주기태스크들의 스케줄링 알고리즘인 RM 스케줄링 알고리즘[1]을 비주기 태스크에 확장하여 정의하였다. 식(1)에서 $S_{(t)}$는 임의의 시점 t를 기준으로 t이전에 도착한 태스크 중 아직 종료시한이 지나지 않은 태스크 집합을 나타내며
$S_{(t)} = \{T_i | A_i \le t < A_i + D_i\}$ 로서 표현된다. 또한 임의의 시점 t에서의 합성 이용률의 상한값은 다음과 값이 정의된다.

$$\begin{cases} UB_{(n)} = \frac{1}{2} + \frac{1}{2n} & n < 3 \\ UB_{(n)} = \frac{1}{1+\sqrt{\frac{1}{2}(1-\frac{1}{n-1})}} & n \geq 3 \end{cases} \quad (2)$$

여기서 n은 현재요청집합(current invocation)에 속하는 비주기 태스크의 개수이며, n이 무한히 증가하게 되면 합성 이용률의 상한값은 $\frac{1}{1+\sqrt{1/2}} \approx 0.59$에 수렴하게 된다.

[8]에서는 임의의 시점 t에서의 합성 이용율 $U_{(t)} = \sum_{T_i \in S_{(t)}} \frac{C_i}{D_i}$이 상한값인 0.59를 넘지 않으면 비주기 태스크가 스케줄링 가능하다는 것을 증명하였다. 또한 합성 이용률을 이용한 스케줄링 분석은 $O(1)$에 수행할 수 있어 수락제어를 수행할 때 적합하다. 비주기 태스크들의 합성 이용율을 구하기 위해서는 임의의 시점 t에 대한 현재요청 집합 $S_{(t)}$를 먼저 구해야한다.

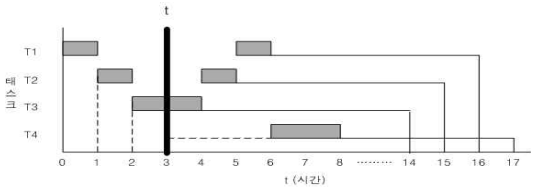


(그림 1) t=7에서의 현재요청집합

(그림 1)에서 임의의 시점 t=7에 대한 현재 요청 집합 $S_{(7)} = \{T_1, T_2, T_4\}$이다. (그림 1)에서 T_3가 $S_{(7)}$에 속하지 못하는 이유는 시점 t=7 보다는 먼저 도착 했으나 종료시한이 t=7 이전인 t=5에 종료되었으므로 $S_{(7)}$에 속하지 않고 합성 이용률을 계산할 때 실행 시간은 포함하지 않게 된다. 다음은 합성 이용률을 구하는 예제이다.

<표 1> 합성 이용율을 위한 비주기 태스크 집합

	A_i	C_i	D_i
T_1	0	2	16
T_2	1	2	14
T_3	2	2	12
T_4	3	2	14



(그림 2) t=3에서의 합성 이용율

합성 이용률을 구하기 위해 먼저 현재 요청 집합을 구하면 $S_{(3)} = \{T_1, T_2, T_3, T_4\}$이고, 이것을 식(1)에 적용하게 되면 $U_{(3)} = \frac{2}{16} + \frac{2}{14} + \frac{2}{12} + \frac{2}{14} = 0.57$ 이다.

$U_{(3)} = 0.57 < 0.59$ 이므로 시점 t=3에 스케줄링 가능하다. 즉, 태스크 T_4가 시점 t=3에 도착했을 때 이미 도착된 모든 태스크들의 스케줄링 가능성을 보장하므로 T_4는 수락된다. 또한 t=4의 시점에서 합성 이용률을 구하게 되면 집합 $S_{(4)}$도 변화가 없고 $U_{(4)}$의 값도 변화가 없으므로 모든 태스크들은 스케줄링 가능하다.

하지만 모든 태스크들의 종료시한이 1만큼씩 줄어들게 되면 $U_{(4)} = \frac{2}{15} + \frac{2}{13} + \frac{2}{11} + \frac{2}{13} = 0.62$ 이 되므로 상한 값 0.59를 넘게 되어 스케줄링 불가능한 상태가 된다. 이 때, $S_{(4)}$의 태스크 중 태스크 T_3는 실행시간이 종료되었지만, 종료시한을 넘지 않았기 때문에 실제 시점 t=4이후에는 실행되지 않더라도 집합 $S_{(4)}$에 속하게 되고 이용률 계산 시 C_4값이 불필요하게 합성 이용률에 함께 계산 된다. 또한 t=4 이후에는 실행시간이 $C_1 = 1$, $C_2 = 1$만큼만 실행하면 되지만, 합성 이용률에서는 $S_{(t)}$에 속하게 되면 모든 실행 시간을 포함하여 이용률을 계산하게 된다.

이러한 문제점으로 인해 특정시간 t에서의 합성 이용률은 계산할 때 t이후에 실제 스케줄링 가능하지만, 스케줄링 불가능하게 판단되는 경우가 발생하게 된다. 본 논문에서는 이러한 문제를 개선하여 스케줄링 가능성을 높이고자 한다.

2.2 개선된 합성 이용율

임의의 시점 t에서 합성 이용률을 계산할 때 먼저 고려되어야 할 것은 시점 t이전에 도착하였으나 아직 시점 t이전에 종료시한을 넘기지 않은

비주기 태스크들의 집합, 즉, $S_{(t)}=\{T_i|A_i \le t < A_i+D_i\}$로서 표현 되는 현재 요청 집합 $S_{(t)}$의 원소에 해당하는 비주기 태스크들을 찾는 것이다. 이러한 $S_{(t)}$에 속하는 비주기 태스크들 중 시점 t를 기준으로 먼저 도착 하였고, 종료시한은 넘지 않았지만, 시점 t이전에 실행시간을 모두 마친 태스크들이 포함될 수 있다. 이러한 태스크들은 임의의 시점 t이후부터 태스크의 종료시한까지 실제 프로세서를 이용하지 않지만 이용률 계산 시 실행 시간이 포함되는 문제점을 가지고 있다. 본 논문에서는 이러한 조건을 제거하여 합성 이용률을 계산한다. 개선된 현재 요청집합을 $S'_{(t)}=\{T_i|A_i \le t < A_i+D_i, t < C_{ie}\}$로 표현하고 여기서 C_{ie}는 비주기 태스크 T_i의 실행시간 C_i의 종료시점이다. C_i의 종료시점은 고정 우선순위를 이용하기 때문에 계산해낼 수 있다.

또한 합성 이용률 $U_{(t)}=\sum_{T_i \in S_{(t)}} \frac{C_i}{D_i}$에서 C_i와 D_i값은 태스크 T_i가 $S_{(t)}$의 조건을 만족하여 $S_{(t)}$의 원소가 되면 항상 동일한 값을 가진다. 즉 시점 t가 증가하더라도 $S_{(t)}$에 속하면 항상 동일한 합성 이용율값을 가지게 되는데 이것은 시점 t이전에 이미 실행 완료된 태스크의 실행시간 값까지 합성 이용률에 포함되기 때문에 실제 남은 실행시간 값이 매우 작더라도 시점 t에서 스케줄링이 불가능하게 판단되는 경우가 발생할 수 있다. 이러한 문제를 개선하기 위해 C_i값과 D_i값을 시점 t를 기준으로 하여 변경한 C_i와 D_i값을 이용하여 합성 이용률을 구한다.

시점 t를 기준으로 하여 태스크 T_i의 실행을 마치기 위해 필요한 최대 수행시간[9]를 잉여 수행시간 $e_{i,t}$라 하고, 시점 t를 기준으로 하여 태스크 T_i의 절대 종료시한 d_i와 현재 시점 t와의 차이, 즉 d_i-t를 리드시간(lead time)[10] $D_{i,t}$라고 정의하면 임의의 시점 t에서의 합성 이용률은 다음과 같이 새롭게 표현할 수 있다.

$$U_{(t)}=\sum_{T_i \in S'_{(t)}} \frac{e_{i,t}}{D_{i,t}} \qquad (3)$$

다음은 새롭게 정의된 $S'_{(t)}$, $e_{i,t}$, $D_{i,t}$를 이용해서 개선된 합성 이용률을 구하는 예제이다.

<표 2> 개선된 합성 이용율을 위한 비주기 태스크 집합

	A_i	C_i	D_i	d_i
T_1	0	2	16	16
T_2	1	2	14	15
T_3	2	2	12	14
T_4	3	2	14	17

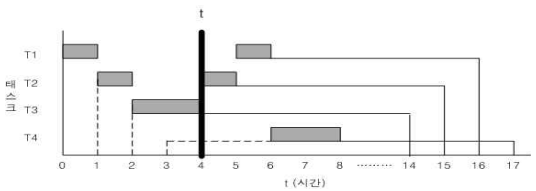
임의의 시점 t=3인 경우에는 개선된 합성 이용율 방법을 적용하더라도 개선된 현재 요청 집합은 $S'_{(3)}=\{T_1,T_2,T_3,T_4\}$으로 기존 방법의 현재 요청 집합과 동일하다. 하지만 개선된 합성 이용율을 구하기 위해 t=3에서 각 태스크들의 $D_{i,t}=d_i-t$와 $e_{i,t}$를 구하여 식(3)에 적용하면,

$$U_{(3)}=\frac{e_{1,3}}{D_{1,3}}+\frac{e_{2,3}}{D_{2,3}}+\frac{e_{3,3}}{D_{3,3}}+\frac{e_{4,3}}{D_{4,3}}$$

$$=\frac{1}{16-3}+\frac{1}{15-3}+\frac{1}{14-3}+\frac{2}{17-3}=0.394$$

이다.

$U_{(3)}=0.394<0.59$ 이므로 임의의 시점 t=3에서 스케줄링 가능하다. 즉, 태스크 T_4가 시점 t=3에 도착했을 때 이미 도착된 모든 태스크들의 스케줄링 가능성을 보장하므로 T_4는 수락된다.



(그림 3) t=4에서의 개선된 합성 이용율

반면 (그림 3)에서처럼 시점 t=4에서 개선된 현재 요청 집합은 $S'_{(4)}=\{T_1,T_2,T_4\}$이 된다. T_3이 개선된 현재 요청 집합에서 제외된 이유는 t=4에서 종료시한은 지나지 않았지만, 실행시간이 모두 종료되었기 때문이다. 식(3)을 이용하면 t=4에서의 개선된 합성 이용율은

$$U_{(4)} = \frac{e_{1,4}}{D_{1,4}} + \frac{e_{2,4}}{D_{2,4}} + \frac{e_{4,4}}{D_{4,4}}$$

$$= \frac{1}{16-4} + \frac{1}{15-4} + \frac{2}{17-4} = 0.328$$

이다.

$U_{(1)}, U_{(2)}$의 경우에는 $S'_{(t)}$ 값의 변화는 없으나 시간의 흐름에 따라 이용율이 변하게 되고, $U_{(5)}, U_{(6)}, U_{(7)}$의 경우는 $S'_{(t)}$ 값도 변하고, 이용율도 변하게 된다. 다음은 식(3)을 이용해 $U_{(1)}, U_{(2)}, U_{(5)}, U_{(6)}, U_{(7)}$을 각각 계산한 결과 이다.

$$U_{(1)} = \frac{1}{16-1} + \frac{2}{15-1} + \frac{2}{14-1} + \frac{2}{17-1} = 0.488$$

$$U_{(2)} = \frac{1}{16-2} + \frac{1}{15-2} + \frac{2}{14-2} + \frac{2}{17-2} = 0.448$$

$$U_{(5)} = \frac{1}{16-5} + \frac{2}{17-5} = 0.257$$

$$U_{(6)} = \frac{2}{17-6} = 0.181$$

$$U_{(7)} = \frac{1}{17-7} = 0.1$$

$U_{(1)}$부터 $U_{(7)}$까지의 합성 이용율을 비교해 보면 $U_{(1)}$보다 $U_{(7)}$이 더 낮은 이용율을 보이는데 이것은 이용율과 비례관계에 있는 실행 시간이 줄어들기 때문이다. 다른 비주기 태스크가 도착하여 수락 여부를 판단하는 경우에 시점 t=3 보다 시점 t=4에서 비주기 태스크가 수락될 가능성이 높아진다고 볼 수 있다.

예제처럼 재 정의된 $S'_{(t)}$, $e_{i,t}$, $D_{i,t}$를 이용하여 합성 이용률을 계산한 후 합성 이용률의 상한 값인 0.59와 비교하여 임의의 시점 t에 도착한 비주기 태스크의 수락 여부나, 이미 도착되어 실행 중인 비주기 태스크들의 합성 이용율을 이용하여 스케줄링 가능성 여부를 판단한다.

다음 3장에서는 기존의 방법과 제시된 개선된 방법을 이용하여 합성 이용률을 계산하고 모의실험을 통하여 비주기 태스크 집합들의 스케줄링 가능성 여부를 판단해 본다.

3. 모의 실험

3.1 모의 실험 방법

본 장에서는 모의실험을 통해 기존의 합성 이용율과 본 논문에서 제시한 방법을 비교한다. 모의실험에서는 도착시간(A_i), 수행시간($C_i > 0$), 상대적 종료시한($D_i > 0$)을 가지는 비주기 태스크 1000개를 랜덤하게 생성하고 임의의 시점 t에서의 기존의 합성 이용률과 본 논문에서 제시한 방법으로 구한 합성 이용률 값을 비교 하였다.

또한 아래와 같이 워크로드와 밀도를 변화시키고, 기존의 합성 이용률과 개선된 방법에서의 이용률 변화, 그리고 각 워크로드 별 밀도에 따른 이용률의 변화를 그래프로서 비교 하였다.

입력 워크로드는 모든 도착된 태스크들의 수행시간의 합과 태스크들이 도착한 시간구간의 비율로 표현할 수 있는데 실험 범위는 50% ~ 150%까지 10%씩 증가되는 시점에서 태스크들의 밀도에 따라 이용율을 측정 하였다. C_i / $\min(d_i, T_i)$로 표현되는 밀도는 0.01, 0.03, 0.05, 0.1, 0.2, 0.4인 태스크들로 구성하여 이용율을 측정하였다.

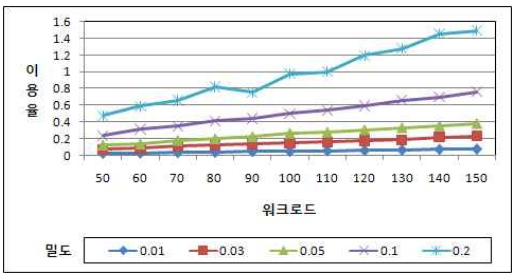
3.2 실험 결과

다음의 (그림 4)과 (그림 5)는 기존의 합성이용률과 개선된 방법을 적용한 합성이용률에 대한 밀도와 워크로드에 따른 비주기 태스크들의 프로세서 이용율을 보여주고 있다.

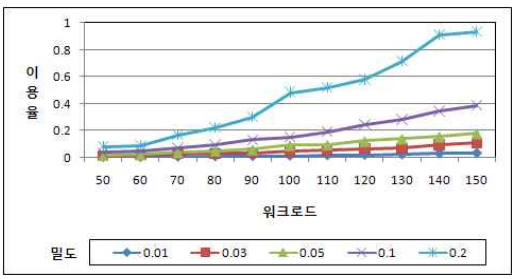
합성 이용률에 의한 방법은 비주기 태스크들의 상대종료시한이 수행시간에 비하여 긴 경우인 밀도가 낮은 경우에 효율적인 결과를 보여준다. 또한 입력 워크로드가 클수록 프로세서 이용률이 올라간다. 그 이유는 입력워크로드가 크다는 의미는 시간구간에 비해 도착되는 비주기 태스크들이 많이 도착된다는 의미이므로 프로세서 이용률이 올라가게 된다. 또한 밀도가 클수록 프로세서 이용률도 올라가게 되는데, 밀도값이 클수록 상대적으로 수행시간($C_i > 0$)보다는 상대적 종료시간($D_i > 0$)이 짧기 때문이다. 즉 이용율과 비례관계인 실행 시간보다 반비례 관계인 종료시한이 작은 값을 가지므로 이용율은 크게 나타난다.

실험결과에서도 볼 수 있듯이 기존 합성 이용율과 개선된 합성 이용율을 비교해 보면 개선된 방법의 이용율이 좀더 낮은 이용율을 갖는다는 것을 알 수 있다. 특히 밀도가 큰 태스크 집합에서의 이용율의 차이가 더 크게 나타남을 알 수

있다.



(그림 4) 합성 이용율



(그림 5) 개선된 합성 이용율

(그림 5)에서 보는 개선된 합성 이용율은 (그림 4)의 합성 이용율과 비교할 때 워크로드와 밀도의 변화에 따른 프로세서의 이용률의 차이가 크지 않게 나타난다. 그 이유는 기존의 합성 이용률에서는 임의의 시점 t에서 이용률을 측정할 때 태스크의 종료시한을 넘기지 않았다면 태스크의 실행이 완료 되었다 하더라도 실행시간을 이용률에 포함시키기 때문이다. 개선된 방법에서는 남은 실행시간만을 이용하여 프로세서 이용률을 계산하기 때문에 실제 임의의 시점 t에서 기존의 합성이용률보다 개선된 합성이용률의 값이 낮은 값을 가지는데, 이것은 이미 입력된 비주기 태스크들의 스케줄링을 보장하면서 시점 t에 입력되는 다른 비주기 태스크들의 수락율을 높일 수 있다는 의미를 가진다.

(그림 6), (그림 7), 그리고 (그림 8)는 기존의 합성 이용률을 이용한 프로세서 이용률과 개선된 방법을 통한 프로세서 이용률의 워크로드별 변화를 보여준다. 그림에서 보듯이 워크로드별로 비교해 볼 때 개선된 스케줄링 방법이 기존의 방법보다 높은 밀도를 가지는 태스크 집합들을 스케줄링 가능하도록 한다는 것을 알 수 있다.

(그림 6) 워크로드가 50% 일때 합성 이용율

(그림 7) 워크로드가 100% 일때 합성 이용율

(그림 8) 워크로드가 150% 일때 합성 이용율

4. 결론

본 논문에서는 시점 t에서 현재 현재요청집합의 조건을 변화시켜 합성 이용율에 의한 프로세서 이용율을 개선함으로써 기존의 합성이용률을 이용하면 스케줄링이 불가능하던 비주기 태스크들을 스케줄링 가능하게 하는 방법을 제시하였다. 또한 실험을 통해 개선된 방법의 워크로드와 밀도의 변화에 따른 프로세서 이용률이 기존의 방법보다 낮은 값을 갖는다는 것을 확인하였다.

이러한 결과는 합성 이용률의 상한값인 0.59를 넘지 않는 범위에서 더 많은 비주기 태스크들을

수락할 수 있다는 의미를 가진다.

일반적으로 태스크들의 수락율이 높을수록 프로세서 이용률은 높아지게 된다. 또한 프로세서 이용률이 같은 경우에는 낮은 밀도를 가지는 태스크들이 높은 밀도를 가지는 태스크 보다 태스크 수락율이 높게 나타난다. 또한 밀도가 높은 태스크들과 낮은 밀도를 가지는 태스크들의 프로세서 이용률을 비교했을 때 개선된 방법에서 태스크들의 밀도가 커지더라도 프로세서 이용률이 증가되는 값의 차이가 작게 나타남을 알 수 있다.

이것은 밀도에 따라 프로세서 이용률의 증가가 크게 변화하지 않기 때문에 밀도가 높은 태스크들이 도착 했을 때 기존의 방법보다 태스크들의 수락율을 높일 수 있다는 의미를 가진다.

본 논문에서는 비주기 태스크만을 고려하여 임의의 시간에서의 합성이용률을 통하여 스케줄링의 가능성 여부를 판단하였지만, 주기 태스크들과의 혼합된 태스크 집합에서의 이용율 측정을 통한 스케줄링 가능성 판단 여부 및 수락율에 대한 연구가 필요하다.

참 고 문 헌

[1] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in hard real time environment", Journal of the ACM, vol. 20, pp. 46-61, Jan. 1973.

[2] H. Chetto and M. Chetto, "Some results of the earliest deadline first scheduling algorithm", IEEE Transactions on Software Engineering, vol. 15, no. 10, pp. 1261-1268, Oct. 1989

[3] J. P. Lehoczky, L. Sha, and J. K. Strosnider, "Enhanced aperiodic responsiveness in hard real-time environments", in Proceedings of the Real-Time Systems Symposium, pp.261-270, Dec. 1987.

[4] B. Sprunt, J. P. Lehoczky, and L. Sha, "Exploiting unused periodic time for aperiodic service using the extended priority exchange algorithm", in Proceedings of the Real-Time Systems Symposium, pp. 251-258, Dec. 1988.

[5] B. Sprunt, L. Sha, and J. P. Lehoczky, "Aperiodic task scheduling for hard real-time systems", The Journal of Real-Time Systems, vol. 1, pp. 27-69, Dec. 1989.

[6] J. P. Lehoczky, and S. Ramos-Thuel, "An optimal algorithm for scheduling soft-aperiodic tasks in fixed-priority preemtive systems", in Proceedings of the real-Time Systems Symposium, pp. 110-123, Dec. 1992.

[7] S. R. Thuel and J. P. Lehoczky, "Algorithms for scheduling hard aperiodic tasks in fixed-priority systems using slack stealing", in Proceedings of IEEE Real-Time Systems Symposium, Dec. 1995, pp. 22-33

[8] T. F. Abdelzaher, V. Sharma, and C. Lu, "A Utilization bound for aperiodic tasks and priority driven scheduling", IEEE Transactions on Computers, vol. 53, no. 3, pp. 334-350, Mar 2004

[9] J. Park, M. Ryu, and S. Hong, "Deterministic and statistical admission control for QoS-aware embedded systems", Journal of Embedded Computing, vol. 1, no. 1, 2004,

[10] J. P. Lehoczky, "Real-time queueing theory", in Proceedings of IEEE Real-Time Systems Symposium, Dec. 1996, pp. 186-195

[11] Kim, In-Guk, Kim, Dong-Yoon and Hong, Man-pyo, "Real-time scheduling of tasks that contain the external blocking intervals", Proceedings Second International Workshop on RTCSA '95, pp. 54-59, 1994.

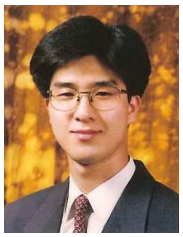
[12] A. Wellings, M. Richardson, A. Burns, N. Audsley, K. Tindell, "Applying new scheduling theory to static priority preemptive scheduling." Report RTRG 1921120, Dept. of Computer Science, Univ. of York.

[13] T. F. Abdelzaher and C. Lu, "Schedulability analysis and utilization bounds for highly scalable real-time services", in Proceedings of IEEE Real-Time Technology and Applications Symposium, May 2001

[14] T. F. Abdelzaher and B. Anderson, J. Jonsson, V. Sharma, and M. Nguyen. "The aperiodic multiprocessor utilization bound for liquid tasks." in Real-time and Embedded Technology and Applications Symposium, San Jose, California, September 2002.

[15] T. F. Abdelzaher and V. Sharma. "A synthetic utilization bound for aperiodic tasks with resource requirements." in 15th Euromicro Conference on Real-Time Systems, porto, Portugal, July 2003.

문 석 환



2001년 : 단국대학교대학원 전자계산학과(이학석사)
2006년 : 단국대학교대학원 전자계산학과(박사 수료)
2006년~현 재 : 영동대학교 임베디드소프트웨어 학과 강의전담 전임교수
관심분야 : 운영체제, 실시간시스템, 임베디드시스템

김 인 국

1982년 : 단국대학교(학사)
1985년 : 미국 에모리대학교(석사)
1995년 : 아주대학교(박사)
1986년~현 재 : 단국대학교 컴퓨터학부 컴퓨터과학전공 교수
2001년~2003년 : 미국 뉴멕시코공과대학 방문교수
관심분야 : 운영체제, 실시간시스템, 임베디드시스템