제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

실시간 시스템에서 주기 및 비주기 태스크의 효율적인 스케줄링 연구

고국희, 이성준, 안광선*
*경북대학교 컴퓨터공학과
e-mail : gogooki@knight.ce.knu.ac.kr

A Efficient Scheduling Study about Periodic and Aperiodic Task on Real-Time System

Guk-Hee Ko, Sung-Jun Lee, Kang-Sun Ahan
Dept. of Computer Engineering, Kyung-Pook University

요 약

실시간 시스템에서 주기 태스크의 마감시간을 보장하며 도착 시간을 예측할 수 없는 비주기 태스크를 처리되도록 스케줄링 하는 것은 매우 중요하다. 본 논문에서는 정적 우선순위 스케줄링 방식 중 최적인 RM 방식으로 주기 태스크를 스케줄링 하고, 비주기 태스크의 발생시 비주기 태스크를 EDF 방식으로 스케줄링하여 오프라인에서 구한 슬랙을 비주기 태스크에 할당한다. 제안한 방식은 비주기 태스크의 마감시한내의 슬랙과 비주기 태스크의 실행 시간을 비교하여 비주기 태스크가 마감시한내에 실행되지 못하는 경우 주기 태스크와 슬랙을 최적적합으로 교환 하는 방식을 사용하여 비주기 태스크의 처리를 가능하게 하였다. 제안된 방식은 경성 비주기 태스크와 연성 비주기 태스크에 적용이 가능하며, 실험 결과 교환 알고리즘을 적용한 스케줄링 방식이 적용하지 않은 스케줄링 방식에 비해 비주기 태스크의 처리율이 높게 나왔다.

1. 서론

실시간 시스템이란 시스템 각 태스크의 처리 마감시간 요구를 만족하여 예측이 가능하도록 하는 시스템을 말한다. [1] 실시간 시스템은 항공기, 미사일, 원자력 발전소, 공장 제어 등 많은 분야에서 사용되고 있으며 최근에는 가전제품이나 휴대용 단말기에도 이용되고 있다.

시스템의 태스크의 처리 마감 시간 요구를 만족하지 못하는 경우 태스크의 실행 결과의 가치가 0 이하이면 경성 실시간 시스템(hard real-time system)으로 분류하며, 가치가 점차 감소하는 경우 연성 실시간 시스템(soft real-time system)으로 분류 한다. [2]

태스크는 성격에 따라 주기적 태스크와 비주기적 태스크로 나눠진다. 주기적 태스크는 고정된 주기에 따라 수행되며 시작시간, 주기, 실행시간, 마감시간의 파라메터로 표현되며 $\tau_i=(\phi\ i, p_i, e_i, d_i)$ 로 표기한다. 주기 태스크의 스케줄링은 주로 고정 우선순위 스케줄링과 동적 우선순위 스케줄링 방식이 사용된다. 고정 우선순위는 태스크마다 정해진 우선순위를 계속 유지하는 것이며, 동적 우선순위는 실행시 우선순위가 정해지며 동적으로 변한다.

비주기적 태스크는 특정 이벤트의 발생에 의해서만 활성화되는 작업이다. 비주기적 태스크는 주기가 알려져 있지 않으며, 시작시간, 실행시간, 마감시간의 파라메터로 표현되며 $Ai = (\phi\ i, e_i, d_i)$로 표기한다. 비주기적 태스크의 스케줄링은 특정 서버를 사용하거나 슬랙 스틸링 방식을 사용한다. 기존에는 주기 태스크와 비주기 태스크의 스케줄링 시 연성 비주기 태스크의 응답시간 개선에 초점을 맞추어 연구를 진행해 왔다.

본 논문은 주기 및 비주기 태스크의 효율적인 스케줄링을 위해 주기적 태스크는 RM(Rate Monotonic)방식을 사용하여 예측성을 가진 스케줄링을 하도록 하였고, 비주기 태스크는 EDF(Earliest Deadling First)방식을 사용하여 마감시간을 보장하는 스케줄링을 하도록 한다. 오프라인에서 초월주기만큼의 슬랙테이블을 만들어 비주기 태스크가 발생하면 이 테이블의 슬랙과 비

주기 태스크의 실행 시간을 비교하여 스케줄링 한다.

다음 장에서는 기존의 관련된 연구에 대해 살펴보고, 3 장에서는 제안하는 스케줄링 알고리즘에 대해 설명하며 4 장에서는 실험을 통해 제안된 방식을 검증하고 5 장에서는 결론을 기술한다.

2. 관련 연구

본 장에서는 주기 태스크와 비주기 태스크를 하나의 프로세서에서 스케줄링 하는 기존의 연구에 대해 살펴본다.

RM 방식으로 주기 태스크를 스케줄링 할 경우 정적으로 할당된 주기 태스크의 우선순위가 비주기 태스크의 발생으로 인해 주기 태스크가 긴급도 변화에 대응하지 못하여 마감시한을 넘기게 된다. 그래서 주기 태스크를 위한 서버를 두는 방법이 연구 되었다.

폴링(Polling)서버 방식은 비주기적 태스크의 실행을 전담하는 폴링서버를 두는 방법인데 비주기 태스크를 위한 주기 기간 동안 비주기 태스크가 발생하지 않으면 그 주기동안의 시간이 낭비가 된다. 대역폭(Bandwidth preserving)서버방식은 폴링 서버의 문제점을 개선하여 서버 주기 동안 비주기 태스크가 발생하지 않으면, 그 시간을 주기 태스크에게 주어 서버의 낭비를 줄이도록 한다. 이러한 방법에는 DS(Deferable Server), Sporadic Server 가 있다.[3][4] 이러한 방법들은 비주기 태스크의 발생률이 낮을 때 비 효율적이다.

EDF 방식으로 주기적 태스크와 비주기적 태스크를 스케줄링 하기 위한 방법은 Chetto 와 Schwan 에 의해 연구 되었다. [5][6] Chetto 는 주기 태스크들이 EDF 에 의해 스케줄 될 때의 경성 비주기 작업들의 수용 알고리즘을 제안하였으나 비주기적 태스크에 대한 실행 요청이 그 전에 실행 중이던 비주기적 태스크들의 실행 종료시점에서만 이루어지며 주기와 마감시간이 일치하는 경우에만 적용 가능하다.

3. 스케줄링 알고리즘

본 장에서는 주기 태스크 스케줄링 방식과 비주기 태스크 스케줄링 방식에 대해 알아보고, 제안된 방식의 알고리즘에 대해 설명한다.

3.1 주기 태스크 스케줄링

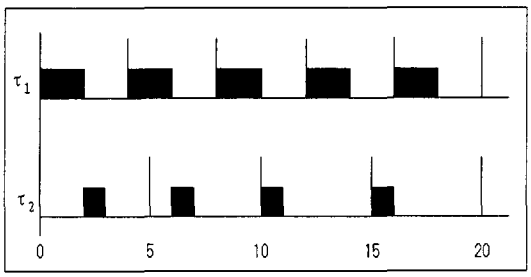
주기 태스크는 주기에 따라 태스크에 고정된 우선순위를 할당하는 RM 방식으로 스케줄링한다. RM 방식은 고정 우선 순위 스케줄링 방법 중 최적으로 증명되었다. [7] 짧은 주기의 태스크에 높은 우선 순위를 부여하며, 실행 중에 더 높은 우선순위를 가지는 태스크가 도착하면 새로 스케줄링을 하여야 한다. 이 방식은 다음과 같은 가정을 하고 있다.

①태스크들은 주기적이고, 주기와 마감시간은 같으며, 실행 중에 스스로 중지 되지 않는다.

②태스크들은 선점 가능하고, 문맥교환과 태스크 스케줄링에 드는 비용은 실행 시간에 포함된다.

③모든 태스크들은 서로 독립적이다.

그림 1 은 RM 방식의 스케줄링 방식을 보여준다. 주기 태스크 집합 τ_1=(0,4,4,2) τ_2=(0,5,5,1) 의 RM 방식 스케줄링 결과이다. 태스크 집합 τ 의 활용률(utilization)은 $U_\tau = \sum_{T_i \in T} \frac{e_i}{p_i}$ 공식을 이용하여 $U_\tau = 0.6$ 을 구할 수 있다.



(그림 1) RM 방식 스케줄링

그림 1 에서 τ_1는 τ_2보다 주기가 짧으므로 우선순위가 높다. 이와 같이 RM 방식은 태스크가 어느 시점에서 실행되고, 마치는지 알 수 있어서 예측성이 좋으며 EDF 방식에 비해 오버헤드가 작다. 그래서 본 논문에선 이 방식으로 주기 태스크를 스케줄링 한다.

3.2 비주기 태스크 스케줄링

EDF 방식은 동적 우선 순위 스케줄링 방법 중 최적으로 알려져 있다. 고정우선순위 방식에서 서버를 두는 방법은 비주기 태스크의 발생률이 낮을 때 비효율적이므로 본 논문에서는 슬랙 스틸링 방식을 사용한다. 비주기 태스크가 발생하면 비주기 태스크를 EDF 방식으로 스케줄링 하여 주기 태스크가 사용하지 않는 여유시간을 비주기 태스크에 할당한다.

슬랙은 주기 태스크의 실행 시 생기는 여유시간을 말하며, 슬랙 스틸링 방식은 슬랙을 계산하여 이 시간을 비주기 태스크 실행에 제공하는 방식으로 최소의 응답을 제공하는 가장 좋은 알고리즘으로 알려 져 있다. [8][9] 이 방식은 태스크들 주기의 최소 공배수인 초월주기(H)를 구해서 이 시간 동안 존재하는 각 태스크의 슬랙을 구한다. 슬랙은 A(i,j) 라는 이차원 배열로 저장된다. i 는 태스크의 우선순위를 나타내고 j 는 태스크의 j 번째 실행을 나타낸다.

그림 1 의 A(i,j)는 A(1,1) = 2, A(1,2) =4 , A(1,3) =6 , A(1,4) =8 , A(1,5) =10 , A(2,1) =1 , A(2,2) =2 , A(2,3) =4 , A(2,4) = 6 이다. 비주기 태스크가 s 라는 시간에 도착했을 때 비주기 태스크를 실행하기 위해 할당할 수 있는 최대 가용 슬랙은 [수식 1]을 이용하여 구할 수 있다.

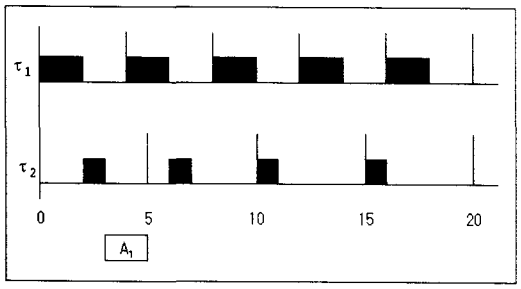
[수식 1] $A^* = \min_{\{1 \le i \le n\}}(A(i,j) - I_i(s) - A(s))$

[수식 1]에서 A(i,j)는 우선순위 레벨 i 에서 비주기 태스크가 처리될 수 있는 가장 큰 시간총합이며 A(s)는 시간 0 에서 t 까지 비주기 태스크를 실행하는데 소비된 누적 시간이며 $I_i(t)$시간 0 부터 t 까지 우선순위 레벨 i 의 슬랙이다.

3.3 제안된 방식

RM 방식은 정적 스케줄링 방식 중 최적이며 EDF 방식은 동적 스케줄링 방식 중 최적이다. 제안된 스케줄링에서는 주기적 태스크를 RM 방식으로 스케줄링 함으로써 예측성을 높이고 오버헤드를 줄이며, 비주기적 태스크를 EDF 방식으로 스케줄링 하여 오프라인에서 구한 슬랙테이블을 이용하여 슬랙을 할당한다.

비주기 태스크의 마감시한까지의 슬랙값과 비주기 태스크의 실행 시간을 비교하여 비주기 태스크의 실행시간이 작거나 같다면 비주기태스크를 실행한다. 비주기 태스크의 실행 시간이 크다면 주기 태스크와 다음 슬랙의 우선순위를 교환하여 슬랙을 먼저 실행 하도록 한다.



(그림 2) 비주기 태스크 스케줄링

그림 2 는 그림 1 의 주기적 태스크가 실행 중에 시간 3 에 비주기 태스크 A_1 이 가 발행하는 경우를 나타낸 것이다. 만약 A_1 의 마감시한이 8 이라면 A_1 은 시간 3 과 7 에서 실행을 할 것이다.

만약 A_1 의 마감시한이 7 이라면 7 까지의 슬랙보다 실행시간이 크므로 비주기 태스크 A_1 은 마감시한을 놓치게 된다. 이 경우 A_1 의 태스크 성격이 경성 비주기 태스크라면 마감시한을 놓치게 되므로 시스템에 치명적인 영향을 미친다. 연성 비주기 태스크일 경우 시스템의 반응시간이 좋지 못하게 된다. 본 논문에서는 경성 비주기 태스크와 연성 비주기 태스크를 함께 고려할 수 있는 스케줄링 기법이다.

비주기 태스크의 마감시한이 7 일 경우 마감시한 7 까지의 슬랙보다 실행시간이 크므로 주기 태스크와 슬랙의 우선순위를 교환하도록 스케줄링한다. 우리는 오프라인에서 구한 슬랙의 값이 3 에서 10 사이의 시간에 2 라는 것을 알 수 있다. 그래서 마감시한을 만족하지 못하는 경우 1 의 슬랙을 사용 후 나머지 슬랙의 우선순위와 주기 태스크 τ_2 의 우선순위를 교환하여 A_1 은 7 에서 종료하고 τ_2 는 7 에서 실행하게 되므로 비주기 태스크의 마감시한을 만족하며 주기 태스크의 실행을 보장한다.

그림 3 은 제안된 방식의 스케줄링 알고리즘이다. 초월주기 동안 비주기 태스크 큐가 비어있지 않고 마감시한까지의 슬랙이 비주기 태스크의 실행시간보다 크거나 같다면 비주기 태스크를 실행한다. 이 때 비주기 태스크는 EDF 방식에 의해 우선순위가 결정된다.

```
Scheduling()
{
 While(current_time = H){
   /*큐안에 비주기 태스크 있고 슬랙 테이블에
    슬랙이 있다면*/
   if (A_Task != 0 && slack_table.slack !=0){
     /*마감시한까지의 실행시간과 슬랙 비교*/
     if (A_Task.실행시간 < slack_table.slack){
       while (slack_table.slack != 0){
         A_Task 실행
         slack_table.slack 갱신
       }
     }
     else
      /*주기태스크와 슬랙 교환 알고리즘 실행*/
   }
   else if{    /*주기 태스크 실행  */
     A_Task 대기
     P_Task 실행  /*주기 태스크 RM방식으로
     스케줄링*/
   }
   else
     Process idle
 }
}
```

(그림 3) 제안된 방식 알고리즘

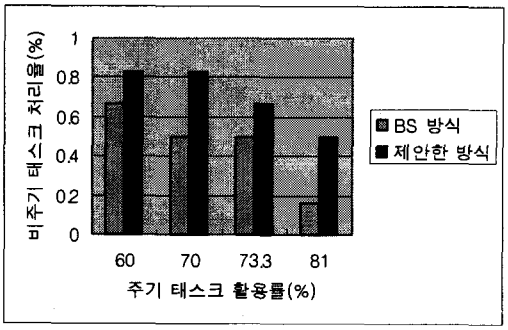
만약 마감시한까지의 슬랙이 비주기 태스크의 실행시간보다 작다면 주기 태스크와 슬랙을 교환한다. 이 알고리즘은 비주기 태스크의 마감시한 안의 주기 태스크와 그 주기 안의 슬랙을 교환하는 방식이다. 이 때, 주기 태스크의 실행크기와 슬랙의 크기가 최적적합(best-fit)인 주기 태스트와 슬랙을 교환한다. 교환 후 실행시간보다 슬랙이 작다면 경성 비주기 태스크이면 스케줄링이 불가능하므로 그에 따른 신호를 프로세서에게 보내고, 연성 비주기 태스크이면 다음 슬랙이 발생할 때까지 대기 한다.

4. 실험 및 실험결과

본 논문의 스케줄링 알고리즘은 c#으로 구현하였으며 정해진 주기적 태스크값과 랜덤한 비주기 태스크 값을 사용하여 실험하였다. 그림 4 는 제안한 방식과 BS(Background Server)방식을 비교한 결과를 보여준다. 제안한 방식과 BS 방식은 주기 태스크의 프로세서 활용률이 각각 60%, 70%, 73.3%, 81% 일 때 비주기 태스크의 처리율은 그림 4 와 같이 보여진다.

제안한 방식이 그림 4 와 같이 BS 방식에 비해 월등히 좋은 비주기 태스크 처리율을 나타내었다. 주기

태스크의 활용률이 60%인 경우 BS(Background Server) 방식과 제안한 방식은 비주기 태스크의 처리율이 큰 차이를 보이지 않았으나, 81% 인 경우 큰 차이를 보였다. 실험 결과 비주기 태스크가 마감시한을 넘기는 경우 마감시한 안의 주기 태스크와 슬랙을 교환 하는 방식이 효과가 있음을 알 수 있다.



(그림 4) 비주기 태스크 처리율 실험 결과

5. 결론 및 향후 과제

실시간 시스템에서 하나의 프로세서에서 주기 태스크와 비주기 태스크의 효율적인 스케줄링을 위해 슬랙을 고려한 방식을 제안한다. 주기 태스크의 마감시간을 넘기지 않는 범위 내에서 시작 시간을 알 수 없는 비주기 태스크의 요청도 처리 할 수 있도록 스케줄링 하는 방식을 제안하였다.

본 논문에서는 정적 우선순위 스케줄링 방식 중 최적인 RM 방식으로 주기 태스크를 스케줄링 하고, 비주기 태스크의 발생시 오프라인에서 구한 슬랙을 이용하여 비주기 태스크를 실행한다. EDF 방식으로 비주기 태스크들을 스케줄링 함으로써 비주기 태스크가 긴급한 태스크를 먼저 처리하였으며 비주기 태스크가 슬랙을 이용하여도 마감시한 내에 처리될 수 없는 경우 주기 태스크와 슬랙 교환 알고리즘을 통해 비주기 태스크의 스케줄링을 가능하게 하였다. 실험 결과 이 방식은 슬랙 교환 알고리즘을 고려하지 않은 방식보다 비주기 태스크의 처리율이 높았다.

앞으로는 RM 방식에서 문맥교환과 오버헤드를 고려한 연구가 필요하며, 멀티프로세서 시스템에서 제안한 방식을 구현하는 연구가 필요하다.

참고문헌

[1] John. A. Stankovic, K. Ramamritham, "Evaluation of a Flexible Task Scheduling Algorithm for Distributed Hard Real-Time Systems," IEEE Transactions on Computers, vol.34, no.2, pp.1130-1143, Dec.1985

[2] Liu. C. L. and Layland. J. W., "Scheduling algorithm for multiprogramming in a hard real-time environment," JACM, Vol.20, pp.46-61, 1773

[3] R.I.Davis, K.W.Tindell, and A.Burns, "Scheduling Slack Time in Fixed Priority Pre-emptive Systems," In Proceedings of the 14th Real-Time Systems Symposium, pp.222-231, 1993

[4] J. P. Ghazalie and T. P. Baker, "Aperiodic Servers in a Deadline Scheduling Environments," The Journal of Real-Time Systems, pp.31--67, 1995

[5] R.I.Davis,K.W.Tindell,and A.Burns, "Scheduling slack time in fixed-priority preemptive systems," in Proceedings of the Real-Time Systems Symposium,pp160-171, Dec.1993

[6] H.Chetto and M.Chetto. "Some Results of the Earliest Deadline Scheduling Algorithm", IEEE Transactions on Software Engineering, vol. 15, no 10, pp1261-1269, 1989

[7] J.Leung and J.Whitehead, "On the complexity of fixed scheduling of periodic real-time tasks," Performance Evaluation, pp253-250, 1982

[8] B. Sprunt, L. Sha, and J. Lehoczky, "Aperiodic Task Scduling for Hard Real-Time Systems," The Journal of Real-Time Systems, pp.251-258, 1989

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