경성 실시간 테스크를 위한 확장된 가능성 검사를 통한 비율단조 기반 스케줄링 기법

신동현, 이준혁, 조수현, 김영학

공과대학교 컴퓨터공학과 대학원

{genius, next, shcho, yhkim}@cespc1.kumoh.ac.kr

The Scheduling Technique Based on Rate-Monotonic with Extended Schedulability Inspection for Periodic Task in Hard Real-Time System

Dong-Hern Shin, Joon-Taek Lee, Soo-Hyun Cho, Young-Hak Kim
Graduate School, Dept. of Computer Engineering, Kumoh National Institute of Technology

요 약

경성 실시간 시스템(Hard Real-Time System)에서는 주기 테스크들의 엉계한 마감시간(Deadline) 보장이 시스템의 성능을 좌우한다. 본 논문에서는 CPU의 이용률(Utility)이 높아 비율단조 정책으로는 마감시간을 보장할 수 없는 주기 테스크들을 위해 확장된 스케줄 가능성 검사를 통해 수행할 테스크들의 공통 주기(L.C.M: Least Common Multiple) 내에서 EDF(Earliest-Deadline First) 정책을 기반으로 마감시간 보장을 수행하기(Peasible Pattern)를 찾아 이를 참조하여 우선순위를 고려하지 않고 테스크들을 강제 수행할 수 있게 하는 비율단조 기반의 스케줄링 기법을 제안한다. EDF를 기반으로 생성된 페턴은 EDF 정책의 특성에 따라 CPU의 이용률을 100% 까지 가능하게 하여 페턴을 참조하여 강제 수행함으로써 EDF 정책이 갖는 실시간 스케줄링 오버헤드를 없앨 수 있다.

1. 서론

실시간 시스템은 동작의 정확성이 높으려는 성능성뿐만 아니라 시간적 정확성에서도 좌우되는 시스템을 의미한다. 실시간 시스템의 중요한 요소는 시간 제약 조건을 마감시간으로 주어진다. 마감시간은 염격하게 있어야 경성이(Hard), 준경성(Firm), 연성(Soft)마감시간으로 분류해 볼 수 있다. 실시간 시스템에 마감시간을 만족시키기 위해서는 고속의 계산을 요구하게 되지만, 고속의 계산이 실시간 시스템의 요구조건을 만족시키는 것은 아니다. 일반적으로 고속의 계산은 시스템의 평균 응답시간을 최소화하기 때문에 실시간 시스템에서 요구되는 수행가능성 즉, 테스크의 마감시간 보장을 의미하지 않는다[1].

본 논문에서는 확장된 스케줄 가능성 검사를 수행하는 비율 단조 기반의 스케줄링 기법을 제안한다. 이 스케줄링은 확장된 스케줄 가능성 검사를 통해 스케줄은 가능하다고 확실하게 마감시간을 보장할 수 없는, 즉, CPU의 이용률 높은 테스크 순서를 찾아내고 검사 시에 이전 마감시간을 만족하는 수행시간을 찾아 이를 참조하여 테스크들을 순서대로 수행할 수 있게 한다. 또한 필요한 오버헤드를 줄이기 위해 비율 단조 방식으로 스케줄이 가능하고 마감시간이 보장하는, 즉, CPU의 이용률 높은 경우 확장 검사를 실시하지 않고 기존의 비율 단조 정책에 따라 각 테스크에 고정 우선순위를 할당하고 그에 따라 수행할 수 있게 한다.

마감시간을 만족하도록 생성되는 수행시간은 EDF 정책을 따르 경우와 동일한 순서가 된다. 따라서 실시간 시스템에 우선순위를 계산하지 않아 스케줄링 오버헤드가 없는 RM 정책의 장점과 높은 CPU 이용률에서도 적용 가능한 EDF 정책의 장점을 모두 수용하고 있다.

2. 관련 연구

2.1 Rate-Monotonic (RM)

비율 단조 알고리즘은 최적의 정적 알고리즘으로 알려져 있고 많이 연구되고 사용되는 방법 중 하나로 다음과 같은 가정 하에 가능하다[2][4].

1) 마감시간을 갖는 모든 테스크들은 주기적이며, 2) 각 테스크에 대한 다음 요청은 이전 요청이 완료된 후 가능하고, 3) 각각의 테스크들은 독립적이다. 4) 각 테스크에 대한 수행시간은 상수이며 수행시간 내에 인터럽트는 없고, 5) 시스템에 있는 다른 비주기 테스크들은 고려하지 않고 주기 테스크에 수행에 드는 비용을 제외한 모든 비용은 무시한다.

특징이라 고정된 각 테스크는 수행 전에 정적으로 우선순위를 부여 받게 된다. 우선순위는 각 테스크의 주기를 기준으로 하며 주기가 짧을수록 높은 우선순위가 주어진다. 다음은 Liu와 Layland에 의해 제안된 스케줄 가능성 검사 방법이다[4].

주기 테스크 집합 \( \tau = \{r_1, r_2, \ldots, r_n\} \) 가 고정 우선순위 방식으로 스케줄링 된다고 가정 할 때, 우선순위 순으로 \( r_1 \)이 가장 높다. 정렬 되었다고 가정한다. \( T_i \)는 \( r_i \)의 주기, \( C_i \)는 수행시간(Completion time), \( C_i/T_i \)는 \( r_i \)의 프로세스당 이용률(Utilization, \( U_i \))과 같다고, 모든 테스크들의 CPU 이용률은 다음과 같다. \( 1 \leq i \leq n \)

\[
U = \frac{C_1}{T_1} + \frac{C_2}{T_2} + \ldots + \frac{C_n}{T_n}
\]

식 (1)의 이용률 \( U \) 가 다음을 만족하면 테스크 집합 \( \tau \)의
모든 테스크들은 동시에 마감시간을 만족할 수 있다.

\[
1 \leq n(2^a - 1) \quad (2)
\]

식 (2)에서 \( n \to \infty \) 이면, \( U = \ln 2 \) (약 0.693)이다. 즉, 주기 기반으로 정적 우선순위를 부여하는 알고리즘에서는 CPU의 이용률이 최대 약 0.693(69.3\%)까지만이다. 따라서, 만약 테스크들의 CPU 이용률이 최대값보다 작으면 마감 시간을 만족하지 못함을 보고 1(100\%)보다 작으면 스키줄링 가능하지만, 마감시간을 만족하지 못할 수 없게 됨은 틀이다. 테스크 설정 전에 우선순위가 할당 되기 때문에 우선순위 계산을 위한 오버헤드가 없고 마감시간과 주기값 간에 적이 적게 됨은 뒤 적이 적다.

2.2 Earliest-Deadline First (EDF)
마감시간 우선 알고리즘은 동적 우선순위 알고리즘으로는 적절한 방법으로 테스크의 주기가 아닌 마감시간을 기 준으로 템의 실행시점에서 실행이 완료되지 않은 테스크들 중에서 가장 가까운 마감시간을 갖는 테스크에게 다음 우선 순위를 할당한다. 즉, 마감시간이 빨라수록 높은 우선순위가 할당된다.

주기 테스크 집합 \( T = \{ T_1, T_2, \ldots, T_n \} \)가 동적 우선순위 방식으로 스키줄링 된다고 가정 할 때, \( T_i \)는 \( T \)의 주기.
\( C_i \)는 수행 시간이라 하면, 다음 조건을 만족할 때 이 테스크들은 스키줄링 가능하다고 할 수 있다. 

\[
\frac{C_1}{T_1} + \frac{C_2}{T_2} + \ldots + \frac{C_n}{T_n} \leq 1 \quad (3)
\]

식 (3)에서 동적 우선순위 알고리즘은 CPU 이용률이 최대 1(100\%)까지 가능함을 알 수 있다. 그러나 새로운 테스크를 할당 할 때마다 우선순위를 계산하기 때문에 실행시간을 스키 줄링 오버헤드가 크다. 마감시간이 주기와 동일하거나 주기 에 비례하여 작을 경우 RM과 같은 마감시간이 작아 보아 작을 때 적이 적다.

2.3 Slack Stealing
슬랙은 CPU의 여유시간을 말한다. 즉, 주기 테스크들의 각 자의 주기 내에서 실행을 완료하고 남은 시간을 말하는데 임 의의 시간대에서 실행되고 있는 주기 테스크들에 없을 때 비주 기 테스크에게 할당하기 위해 주기 테스크로부터 살점을 빼야할 비주기 테스크의 부양시간을 최소화 하고자 하는 방법 이다[5].

3. 제한된 스키줄 가능성 검사 기법
관련 연구를 바탕으로 확장된 스키줄 가능성 검사 기법을 제안한다.

본 논문에서 제안하는 기법은 xi와 같은 제한 사항들을 갖는다. 테스크 수행에 필요한 비용을 제한한 모든 비용은 무시하고, 주기 테스크의 주기는 상수가며, 수행 중 인터럽트는 없고 마감시간은 주기보다 크거나 같다고 가정한다.

이런 조건은 본 논문에서 제한하고 있는 확장된 스키줄 가능 성 검사 기법을 나타내고 있다. 먼저, 임의의 시간대 주기에 테스크 \( T_j \)가 도착하면 \( T_j \)과 동시에 스키줄링이 되어야 할 이전 테스크들 \( T_i \)의 CPU 종 이용률 \( U \)을 구한다. 계산된 \( U \)가 1보다 큰 경우 \( T_j \)을 포함한 모든 테스크들의 \( U \)가 100\%를 넘는다는 것을 의미하므로 \( T_j \)는 스키줄링 불가능하다 는 것을 알 수 있다.

주기 테스크 \( T_n \) 도착, \( (n > 1) \)

\[
U = \sum_{i=1}^{n} \frac{C_i}{T_i} \quad */ 이용률(Utilization) 계산*/
\]

\[
if(U \leq 1) \{
if(U \leq n(2^a - 1)) \{
RM 정책 스케줄링
else \{
가능성 검사 대상 테스크들의 주기 값의 L.C.M 계산 /*공통 반복 주기*/
T = (U x L.C.M) / 100 /*할당 가능한 총시간*/
while(T){
EDF 정책을 이용한 테스크 선택
if(테스크 선택 불가능)
스케줄 불가능, 보류
T = 선택된 테스크의 수행 가능 시간량 /* 선택된 테스크의 수행 가능 시간량 과 테스크 반호들 Waiting에 가장 */
}
else(
스케줄 가능, 보류
}
}
그림 1. 제한된 스키줄 가능성 검사 기법

그러나 \( U \)가 식 (2)를 만족할 경우 완벽하게 RM 정책으로 스키줄링 가능하다는 것을 의미한다. 따라서 \( U \)가 1보다 작고 식 (2)를 만족하지 않는 경우 즉, RM 정책으로는 마감 시간을 보장 받지 못하는 경우이다. 본 논문에서는 이러한 경우를 위해 스키줄 가능성 검사 기법을 확립한다. 먼저 이와 같은 경우에는 RM 정책으로는 마감시간을 보장 받을 수 없지만 \( U \)가 1보다 작기 때문에 EDF 정책을 이용하면 보장 가능성이 커질 수 있다.

따라서 EDF 정책을 이용하여 마감시간을 보장할 수 있는 수행범위와 수행가능 시간량을 계산한다. 그러나 EDF 정책이 실패하는 경우에 스키줄 불가능하게 된다. 계산된 시간 \( U \)은 실제 L.C.M까지의 실제 시간량으로 바꾸고 새로운 테스크와 시간차량의 짧은 테임에 유지 하면서 나머지 시간량이 0이 될 때까지 반복한다. 탐색이 끝나면 각 테스크들의 L.C.M을 공통주기로 하여 반복할 수 있는 마감시간 보장 수
행 패턴이 태블론에 저장된다[6].

이렇게 생성된 패턴을 우선순위로 그대로 적용하여 순차적으로 계산된 시간만큼의 실행된 후, 태블론에서 참조되고 있는 다음 테스크가 자동으로 선택되어 실행된다. 패턴 생성 후 공통 주기에서 실행하면 그 다음 공통 주기내에서 패턴을 그대로 사용할 수 있기 때문에 다른 테스크가 도착하지 않으면 이 이상의 스케줄링 오버헤드는 없다.

그림 2. 생성된 패턴의 반복

<그림 2>는 L.C.M을 주기로 패턴이 반복될 것으로 보아진다. Task 1, 2, 3의 주기와 수행시간은 각각 (50, 25), (75, 30), (150, 5)이다. 이들 3개의 수행시간과 L.C.M은 (150)으로 계산된다. 이들 3개의 수행시간의 총 합은 (150 + 75 + 150) = 375이므로, 3개의 수행시간의 총 합이 L.C.M을 만족한다.

4. 평가

<그림 2>의 빗장 틀 부분은 본 논문에서 제안하고 있는 기법이 적용되는 범위를 나타내고 있다. (식 (2))

그림 3. 제안된 기법의 적용 범위

<그림 3>는 주어진 주기와 수행시간에 따른 테스크들의 스케줄 가능여부를 보여준다.

그림 4. RM 정책 결과

<그림 4>에서 테스크 1, 2, 3은 식 (2)에 의해 마감시간을 보장 받는다. (U = 75%) 그러나 테스크 4가 추가된 경우 보장 받지 못한다. (U = 85%) 따라서 RM 정책 적용 시 <그림 4>의 결과와 같이 실패한다. 마감시간을 보장 받지 못한 경우 제안한 기법에서는 EDF 정책을 적용하여 <그림 5>과 같이 각 테스크들의 마감시간을 보장 할 수 있다. <그림 2>의 경우처럼, 생성된 수행 패턴은 각 테스크 주기를 L.C.M을 공통 주기로 반복 되며, 이는 생성된 패턴을 참조하여 수행 하므로 실시간 스케줄링 오버헤드가 없다.

5. 결론 및 향후 연구 제안

본 논문에서는 RM 정책을 기반으로 하는 임의 우선순위 할당 알고리즘에, 좀 더 정확하고 효율적인 스케줄을 하기 위해 확장된 스케줄 가능성 검사 기법을 제안 하였다. RM 정책으로 마감시간을 보장 할 수 없는 테스크들을 위해 미리 수행 패턴을 생성함으로써 임의 우선순위 알고리즘의 정확성과 신속성을 유지하면서 동시에 임의 우선순위 알고리즘의 높은 이용률을 얻을 수 있다. 그러나 실험적 제한에 가까이 훼손된 실질적인 결과를 얻을 수 있다. 각 테스크의 주기를 L.C.M 값을 이용하여 공통 주기를 찾아 반복 패턴을 찾으면서 스케줄링 오버헤드를 줄이고 이 값이 작 은수록 최적이라 할 수 있다. 하지만 값이 커질수록 추가 오버헤드가 늘어난다. 따라서 실제 에를 가진 시스템을 위한 실험적인 논문이 더 필요할 것으로 생각된다. 또한 다양한 실험 환경에서의 성능평가 또한 이루어져야 할 것이다.

참고 문헌