http://dx.doi.org/10.3745/KIPSTA.2011.18A.4.143

불확정 계산을 위한 EDF 기반의 실시간 스케줄링 알고리즘

최 환 필[†] · 김 용 석[††]

요 약

본 논문은 필수 실행 부분과 선택 실행 부분으로 구성된 불확정 태스크(imprecise task) 모델에서 효과적으로 스케줄링 하는 EDF(Earliest Deadline First)기반의 알고리즘을 제안한다. 이러한 태스크 모델은 태스크가 과부하 상태가 되었을 때 처리하는데 유용하게 사용된다. 과부하 상황이 발생하면 선택 실행 부분 중 일부를 포기해야 하는데, 제안한 DOP 알고리즘은 이후에 발생할 태스크에 대해서 보다 유연하게 대처 할 수 있게 하기 위해서 마감시간이 빠른 태스크의 선택 실행 부분을 제거하고, 마감시간이 늦은 태스크의 선택 실행 부분을 남기는 방법을 사용한다. 시뮬레이션을 통하여 성능을 평가한 결과 DOP는 기존에 연구된 스케줄링 알고리즘들에 비해서 좋은 성능을 보였다.

키워드 : 실시간 시스템, 불확정 계산 모델, 온라인 스케줄링, 비주기적 태스크

An EDF Based Real-Time Scheduling Algorithm for Imprecise Computation

Hwan-pil Choi[†] · Yong-seok Kim[††]

ABSTRACT

This paper presents an EDF based scheduling algorithm for scheduling imprecise computation model where each task consists of mandatory part and optional part. Imprecise computation is useful to manage overload condition. In overload situation, some optional parts should be removed. The proposed DOP algorithm removes optional parts of earlier deadline tasks to enhance flexibly for newly arriving tasks. A simulation result shows that DOP has better performance than other algorithms.

Keywords : Real-Time System, Imprecise Computation Model, On-line Scheduling, Aperiodic Task

1. 서 론

온라인 실시간 시스템은 발생하는 태스크들을 마감시간 이내에 완료해야 하는 시간적 제약조건과 실시간으로 발생하는 태스크들에 대한 사전 정보를 가지고 있지 않기 때문에 각 태스크들이 도착한 시점에 적절한 방법으로 스케줄링을 해야 한다. 각 태스크들을 스케줄링 하는 시간이 길어지면 문제가 발생할 수 있기 때문에 복잡한 알고리즘을 적용하는 것은 다소 무리가 있다.

불확정 태스크(imprecise task)는 필수 실행 부분과 선택 부분으로 구성된 태스크를 말한다. 필수 실행 부분은 결과의 정확도를 판단하기 위해 마감시간 이내에 반드시 완료되어야 하는 것이고, 선택 실행 부분은 태스크 실행 결과의 질을 향상시키기 위해 마감시간 이전에 가능한 많이 처리하는 것이 중요하다. 실시간 시스템에서 태스크들이 항상 일정하거나 적게 혹은 많이 발생하는 것이 아니라 외부 환경에 대한 영향에 따라 달라질 수 있기 때문에 일시적으로 과부하 상황이 발생할 수 있다. 이러한 과부하 상황에서 불확정 계산으로 선택 실행 부분을 일부 포기함으로써 각각의 태스크의 결과의 질을 조금 낮추지만 과부하 상황에서 더 많은 태스크들의 필수 실행 부분을 각각의 마감시간 이내에 처리할 수 있는 방법 등의 불확정 계산 모델에 대한 연구가 되고 있다[1,2,3,4,9-11].

주기적인 태스크들은 이후에 발생할 태스크에 대한 정보를 미리 가지고 있어서 이러한 정보를 이용하여 적절한 방법으로 스케줄링 성공률을 높일 수 있지만, 비주기적인 태스크들은 미래에 발생할 태스크들에 대한 예측을 할 수 없다. 과부하 상황이 아니라면 EDF 등의 증명된 최적 스케줄

† 준 회 원 : 강원대학교 컴퓨터정보통신공학부 석사과정
†† 종신회원 : 강원대학교 컴퓨터학부 교수
논문접수 : 2011년 3월 23일
수 정 일 : 1차 2011년 5월 30일
심사완료 : 2011년 6월 8일

링 알고리즘을 사용하면 모든 태스크를 스케줄링 할 수 있다. 하지만, 과부하 상황에서 이후에 발생할 태스크에 대한 정보가 없어서 현재 선택된 태스크가 항상 최적이라고 할 수는 없기 때문에 최적의 스케줄링 알고리즘이 존재할 수 없다[5].

불확정 태스크에 대한 연구는 이러한 상황에서 보다 중요한 일을 더 많이 처리하여 각 태스크 실행 결과의 질은 떨어지더라도 전체적인 실행 결과의 질을 향상시키는 방법이다. 불확정 태스크는 마감시간 이내에 반드시 실행되어야 하는 필수 실행 부분과 태스크 실행 결과의 질을 향상시키기 위해 실행되지만, 과부하 상황에서 필수 실행 부분의 수행을 위해 일부 포기할 수도 있는 선택 실행 부분으로 나뉘고, 마감시간 이내에 실행되지 못한 선택 실행 부분이 에러가 된다. 이러한 에러는 스케줄링 되지 못한 태스크의 선택 실행 부분도 포함하며 총 에러는 발생한 전체 태스크의 선택 실행 부분에서 처리하지 못한 선택 실행 부분의 합으로 계산한다.

기존에 연구된 NORA 알고리즘은 불확정 온라인 태스크 시스템에서 최소의 총 에러를 가지는 스케줄링이 가능하다고 하지만, 특정한 상황과 제약조건하에서만 최소 에러를 가지게 되며 제약조건 이외의 상황에서는 스케줄링 가능한 태스크 집합을 스케줄링 할 수 없는 상황이 발생할 수 있다[3]. 반면, MF(Mandatory First)의 경우는 필수 실행 부분을 먼저 실행함으로써 뛰어난 필수 실행 부분의 스케줄링 성공률을 보인다. 하지만, 필수 실행 부분이 우선 실행되어 실행할 수 있는 선택 실행 부분에 대한 마감시간을 놓치는 경우가 발생하여 많은 에러가 발생하고, 결과적으로 각 태스크의 실행 결과의 질이 떨어지는 문제가 있다[4].

본 논문에서 제안하는 알고리즘은 이후에 발생할 태스크에 대해 보다 유연하게 대처를 할 수 있게 하기 위하여 마감시간을 기준으로 스케줄링 하되 각 태스크의 최대 선택 실행 부분을 확보하면서 실행 가능한 선택 실행 부분을 마감 시간이 가장 뒤에 있는 태스크들에 할당하여 선택 실행 부분의 실행을 최대한 지연시키도록 한다. 선택 실행 부분의 실행을 지연시킴으로써 이후에 새롭게 발생할 태스크의 필수 실행 부분 수행을 위해 포기할 수 있는 선택 실행 부분의 단위 시간을 높임으로써 새로운 태스크의 필수 실행 부분의 스케줄링 가능성을 높이는 알고리즘을 제안한다.

본 논문의 구성은 2장에서 시스템 모델과 관련 알고리즘들의 특성에 대해 기술한다. 3장에서는 DOP 알고리즘에 대해 설명하며, 4장에서 기존 알고리즘들과 제안한 알고리즘의 성능 평가 결과를 비교하고, 5장에서 결론 및 향후 연구에 대해 설명한다.

2. 관련 연구

실시간 스케줄링 알고리즘으로 잘 알려져 있는 EDF는 마감시간이 빠른 순서로 태스크를 스케줄링 하는 것으로 단일 프로세서 상에서 최적임이 증명되었다[7,8]. 또한 고정 값의 마감시간에 따라 우선순위가 결정되므로 문맥 교환 횟수가 낮으며 단순하여 구현이 용이하다는 장점이 있어 단일 프로세서 상에서 EDF전략을 사용하는 것이 적절하다.

불확정 태스크는 필수 실행 부분과 선택 실행 부분으로 나뉘며 필수 실행 부분은 마감시간 이내에 반드시 완료되어야 하는 것이고, 선택 실행 부분은 필수 실행 부분이 완료된 이후에 태스크 결과의 질을 높이기 위해 수행되는 것이다. 불확정 계산을 함으로써 과부하 상황에서 각 태스크 결과의 질을 낮추지만 중요한 필수 실행 부분을 더 많이 실행함으로써 전체적인 결과의 질을 높인다.

본 논문은 단일프로세서 상에서 비주기적으로 발생하는 경성 불확정 태스크를 다루는 EDF기반의 스케줄링 알고리즘을 제안한다. 선택 실행 부분은 필수 실행 부분의 실행이 완료된 이후에 실행할 수 있으며, 각 태스크는 도착 즉시 실행 가능한 상태가 되는 것을 가정한다. 태스크는 각각의 도착시간, 필수와 선택 실행시간, 마감시간을 가지며 태스크 T_i는 $<R_i, M_i, O_i, D_i>$로 표현한다. 에러는 각 태스크의 마감시간 이내에 실행되지 못한 선택 실행 부분의 합을 나타내고 에러에 대한 각 태스크의 중요도는 동일한 것으로 본다.

NORA 알고리즘은 필수 실행 부분을 보장하기 위해서 예약 리스트를 관리하며 이것은 태스크 집합의 필수 실행 부분을 보장하기 위한 것으로 마감시간이 늦은 태스크의 필수 실행 부분을 뒤쪽에 할당하여 채워나가는 방식을 사용한다. 새로운 태스크가 도착하거나 태스크가 완료되는 시점에 예약 리스트를 갱신하며 관리하며 예약된 부분에 도달할 경우, 현재 처리중인 태스크가 예약되어 있으면 예약을 취소하고 처리중인 태스크를 계속 실행하게 되고, 예약 되어있지 않으면 태스크를 종료시키고 예약 된 태스크를 처리한다.

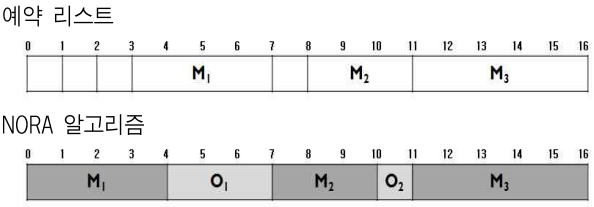
NORA 알고리즘은 불확정 태스크 집합에서 최소의 총 에러를 얻을 수 있다고 알려진 방법이다. 하지만, 선택 실행 부분이 앞에서 실행됨으로써 이후에 발생하는 태스크의 필수 실행 부분을 예약 리스트에 할당하지 못하게 되어 스케줄링 성공률이 떨어지는 문제가 발생한다.

<표 1>의 태스크 집합을 NORA 알고리즘을 적용하여 스케줄링 한 결과는 (그림 1)과 같다. 시간 8에서 태스크 T_4가 발생하였지만, 해당 시점에 1 단위 시간 만큼의 할당이 가능하기 때문에 새로운 태스크의 필수 실행 부분을 예약 리스트에 할당 할 수 없어서 태스크를 처리하지 못하게 된다. 발생한 태스크 집합의 필수 실행 부분은 14 단위 시간이지만, 처리된 필수 실행 부분은 12이고, 총 에러는 처리되지 못한 선택 실행 부분의 4 단위 시간이 된다. 필수 실행 부분의 처리율은 85%이고, 선택 실행 부분의 처리율은 50%가 된다. NORA는 현재 수용된 태스크 집합에 대해서는 최소의 에러를 가지게 된다. 즉, NORA는 스케줄링 가능한 태스크 집합에 대해서는 선택 실행 부분을 최대한 처리할 수 있다. 만약, T_4가 발생하지 않았다면 NORA 알고리즘은 발생한 태스크의 필수 실행 부분을 모두 처리하고 선택 실행 부

분의 처리 또한 최소의 총 에러를 가질 수 있게 스케줄링 하였을 것이다.

〈표 1〉 태스크 집합

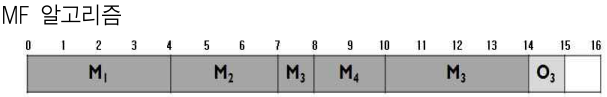
	R_i	M_i	O_i	D_i
T_1	0	4	3	7
T_2	0	3	1	12
T_3	0	5	1	16
T_4	8	2	3	13



(그림 1) NORA 알고리즘 예

MF 알고리즘은 실행할 필수 실행 부분이 있다면 필수 실행 부분을 우선적으로 실행하고 실행할 필수 실행 부분이 없는 경우 선택 실행 부분을 실행하는 것으로 필수 실행 부분에 대해서 높은 스케줄링 가능성을 가지게 된다. 하지만, 필수 실행 부분을 우선 실행함으로써 처리 할 수 있던 선택 실행 부분의 마감시간이 지나 실행할 수 없게 되어 전체적인 결과의 질이 떨어지는 문제가 생길 수 있다.

MF 알고리즘으로 <표 1>의 태스크 집합을 스케줄링 한 결과는 (그림 2)와 같으며 새로운 태스크 T_4는 처리할 수 있었지만, 최선의 경우 1 단위 시간의 선택 실행 부분을 수행할 수 있음에도 불구하고 필수 실행 부분을 우선적으로 실행하게 되어 수행 할 수 있는 선택 실행 부분의 마감시간이 지나게 되어 에러가 높아지게 되는 경우를 보여준다. MF의 경우 발생한 총 에러는 7이며, 필수 실행 부분의 처리율은 100%이고, 선택 실행 부분의 처리율은 12%가 된다. 만약, T_4가 발생하지 않았다면 MF 알고리즘은 M_1을 처리한 후에 M_2를 처리하면서 M_1의 마감시간이 지나게 되고, 이후에 M_3가 처리되면서 M_2의 마감시간 또한 지나가게 되어 최선의 경우 4 단위시간의 선택 실행 부분을 실행 할 수 있는 상황에서 T_3의 1 단위 시간만큼의 선택 실행 부분만을 처리하게 된다.

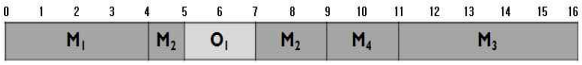


(그림 2) MF 알고리즘 예

<표 1>의 태스크집합을 오프라인으로 처리할 경우, 발생한 태스크의 필수 실행 부분은 모두 처리가 되고 총 에러는 2가 되는 것이 최선의 경우가 된다. NORA 알고리즘은 필수 실행 부분을 예약 리스트에 할당하여 관리함으로써 필수 실행 부분을 보장하면서 수행 가능한 태스크 집합의 최소의 총 에러를 얻을 수 있다. 하지만 선택 실행 부분의 이른 실행으로 새로운 태스크를 처리하지 못하여 필수 실행 부분의 스케줄링 성공률이 낮아지게 된다. MF 알고리즘은 필수 부분을 우선 실행함으로써 높은 필수 실행 부분의 스케줄링 성공률을 가지지만, 태스크 집합의 처리될 수 있는 선택 실행 부분이 제때 처리되지 못하여 마감시간을 놓치기 때문에 에러 또한 높아지게 된다.

DOT 알고리즘[9]은 본 논문에서 선택 실행 부분의 실행을 지연하는 것에 대한 비슷한 아이디어를 가진다. DOT 알고리즘은 각각 필수, 선택 및 필수와 선택에 대한 R(M), R(O), R(M+O) 3개의 예약 리스트를 NORA와 같은 방법으로 관리하여 선택 실행 부분의 여유시간이 0이 될 때, 선택 실행 부분을 처리할 수 있게 한다. 선택 실행 부분을 처리하는 시점을 여유 시간이 0이 되게 함으로써 최대한의 지연을 가능하게 한다. 하지만, 3개의 예약 리스트를 새로운 태스크가 발생하거나 태스크를 처리할 때마다 추가 및 갱신하여 관리해야 하는 문제와 마감시간에 의한 우선순위는 필수 부분에만 적용되고 선택 부분은 마감시간이 빠르더라도 마감시간이 늦은 필수 부분이 먼저 처리되는 문제가 발생하여 태스크 간 선행 제약이 있는 경우에 적합하지 않게 되며, 태스크의 선점이 많이 일어나 빈번하게 문맥교환이 이루어져야 하므로 런타임에서 스케줄링 오버헤드가 크다는 문제가 있다.

DOT 알고리즘으로 <표 1>의 태스크 집합을 스케줄링한 결과는 (그림 3)과 같으며 T_1의 선택 부분을 지연하면서 마감시간이 늦은 T_2의 필수 부분을 먼저 실행하여 추가적인 문맥교환이 발생하게 된다.



(그림 3) DOT알고리즘 예

3. DOP 알고리즘

DOP(Deferable Optional Part) 알고리즘은 과부하 시 전체 태스크의 선택 실행 부분 중에서 많이 지연 가능한 태스크를 우선적으로 남기는 정책을 적용한다. 태스크 집합의 최소 에러를 만족하면서 이후에 발생할 새로운 태스크에 대해 보다 유연하게 처리할 수 있도록 필수 실행 부분을 앞에 실행 될 수 있도록 하고 선택 실행 부분의 실행을 최대한 지연시키게 한다. 선택 실행 부분의 실행을 뒤로 지연시킴으로써 새로운 태스크가 발생 했을 때 지연되어 뒤에 실행하게 한 선택 실행 부분을 새로운 태스크의 필수 실행 부분의 수행을 위해 포기할 수 있게 한다. 따라서 포기할 수 있는 선택 실행 부분의 수행 시간을 높일 수 있고, 새로운 필수 실행 부분의 스케줄 가능성 또한 높아질 수 있다.

선택 실행 부분을 지연시키기 위해 EDF를 기반으로 스케줄링 하면서 전체 태스크 집합에서 처리 할 수 있는 최대

선택 실행 시간을 계산하여 처리할 수 없는 마감시간이 빠른 선택 실행 부분을 제거함으로써 늦은 마감시간을 가지는 태스크에 현재 태스크 집합에서 최대로 가능한 선택 실행 부분을 실행 할 수 있게 한다. 따라서 현재 태스크 집합의 에러를 최소화 할 수 있고, 선택 실행 부분이 뒤에 할당됨으로써 NORA 알고리즘 보다 이후에 발생할 태스크의 필수 실행 부분에 대한 스케줄링 가능성을 높이고, MF 알고리즘 보다 현재 상황에서 가능한 에러를 최소화 할 수 있게 된다. DOT 알고리즘은 선택 부분의 실행을 여유 시간을 고려하여 가능한 많은 지연이 가능하지만, 현재 실행 중인 태스크보다 마감시간이 빠른 태스크가 발생하여 생기는 선점 외에 마감시간에 대한 우선순위에 상관없이 처리되는 과정이 생겨 빈번한 태스크의 선점이 일어나게 된다. DOP 알고리즘은 태스크에 대한 처리 가능한 선택 실행 부분을 마감시간을 기준으로 조절하여 EDF로 스케줄링 하기 때문에 선점 현상은 현재 실행 중인 태스크보다 마감시간이 빠른 새로운 태스크가 발생하는 경우에만 발생하게 되며, 태스크의 지연이 DOT 알고리즘에 비해 약간 적게 지연되는 문제가 생기지만, 이러한 차이는 실험결과 미미한 차이를 보인다.

(그림 4)는 DOP 알고리즘을 의사코드로 표현한 것이다. DOP 알고리즘은 EDF에 불확정 태스크 처리를 위해서 선택 실행 부분을 제거하는 작업을 추가한 것으로, 알고리즘이 단순하며 구현이 용이하다. 새로운 태스크가 발생한 경우 실행할 태스크가 없다면 바로 레디 큐에 넣어 실행 될 수 있도록 한다. 레디 큐는 태스크의 마감시간을 기준으로 빠른 것이 앞쪽에 위치하게 한다. 만약 선택 실행 부분만 남은 태스크와 필수 실행 부분이 남아있는 태스크가 같은 마감시간을 가지고 있는 경우에는 필수 실행 부분이 남아있는 태스크를 앞에 두어 먼저 실행될 수 있게 한다. 레디 큐가 비어 있지 않은 경우에 비주기적으로 발생하는 새로운 태스크들에 대해 현재 상황에서 실행 가능한 태스크를 받아들일지에 대한 수용 테스트(acceptance test)를 하여 통과한 태스크들이 레디 큐에 들어가며, 테스트를 통과하지 못한 태스크는 포기하게 된다. 이 테스트는 현재 검사할 태스크와 이전 태스크의 실행 시간의 합이 현재 검사하는 태스크의 마감시간을 넘어서지 않는지 검사하는 EDF 보장 알고리즘(EDF guarantee algorithm)을 사용한다[6]. 전체 태스크 실행시간이 아닌 필수 실행 부분의 실행시간의 합으로 계산한다.

```
S : set of accepted tasks

if ( a new task T is arrived )
  if( S∪{T} is feasible )
    S ← S∪{T}
    AdjustOptionalPart(S)
  else
    reject T

execute the earliest deadline task of S
```

(그림 4) DOP 알고리즘

```
AdjustOptionalPart(S)
  S is ordered by deadline
  t = current_time();
  for( each task Ti of S in non-decreasing
       order of deadline )
    m = sum of remaining mandatory parts
        of Tj from j=1 to i
    o = sum of remaining optional parts
        of Tj from j=1 to i
    if( (m + o + t) > Di )
      e = (m + o + t) - Di;
      for( each task Tj from 1 to i )
        if( e > oj )
          e = e - oj; oj = 0;
        else
          oj = oj - e; e = 0;
          break;
  return;
```

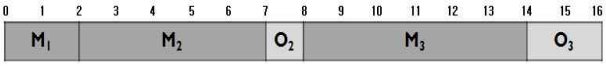
(그림 5) 선택 실행 부분 조절 알고리즘

(그림 5)는 태스크의 선택 실행 부분 조절 알고리즘을 설명한다. 태스크 T_i는 <R_i, M_i, O_i, D_i>로 구성되며 각각 해당 태스크의 도착 시간, 필수 실행시간, 선택 실행시간 및 마감시간을 의미한다. m와 o는 각각 i번째 태스크의 남아있는 필수 실행 부분과 선택 실행 부분의 합을 나타내고, o_j는 j번째 태스크의 남아있는 선택 실행 부분을 나타낸다. e은 제거할 선택 실행 부분의 단위시간을 의미한다. 선택 실행 부분 제거 알고리즘은 레디 큐 내의 태스크를 검사하면서 진행한다. 실제 스케줄링에서는 태스크의 수용 테스트를 진행하면서 수행될 수 있기 때문에 큰 스케줄링 오버헤드 없이 진행할 수 있다. 현재 태스크의 마감시간에서 현재 태스크와 현재 태스크보다 마감시간이 같거나 빠른 태스크의 남아있는 필수 실행 부분의 합을 뺀 값이 해당 상황에서 실행 가능한 최대 선택 태스크의 실행 시간임을 의미한다. 그 값이 현재 태스크와 이전 태스크의 남아있는 선택 실행 부분의 합보다 크다면 모두 실행 가능하다는 것을 의미한다. 그러나 작다면 그 차이만큼을 제거해야 스케줄링 가능하다는 것을 의미하므로 그 차이를 마감시간이 빠른 태스크에서 선택 실행 부분을 제거하여 선택 실행 부분의 실행을 지연시켜 이후 발생할 태스크의 필수 실행 부분의 수행을 위해 포기할 수 있는 선택 실행 부분의 단위시간을 커지게 한다.

(그림 6)은 선택 실행 부분의 지연에 따른 스케줄링 결과를 비교한 것이다. T_1(0, 2, 5, 8), T_2(0, 5, 7, 12)의 태스크가 있을 때 선택 실행 부분을 지연시키지 않은 경우와 선택 부분을 지연시킨 경우를 나타내고 있다. 두 가지 경우 모두 현재 태스크 집합에서 가능한 5 단위 시간만큼의 선택 실행 부분을 처리하게 되고, 이때 새로운 태스크 T_3(8, 6, 2, 16)이 발생하면, 선택 부분을 지연시키지 않은 경우 새로운 태스크의 필수 부분인 6 단위 시간 만큼을 실행 할 수 없어 새로운 태스크를 처리하지 못하게 되고, 선택 부분을 지연시킨 경우는 지연된 T_2의 선택 부분이 4 단위 시간 만큼 포기될 수 있고 새로운 태스크의 필수 부분을 처리하고, 마감시간이 늦은 T_3에서 선택 실행 부분이 실행 할 수 있게 한

다. 이렇게 선택 실행 부분의 지연을 통하여 이후에 발생할 새로운 태스크의 스케줄링 가능성을 향상시키게 된다.

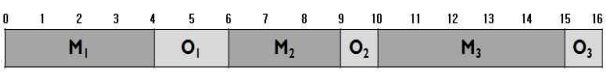
(a) 선택 부분을 지연시키지 않은 경우



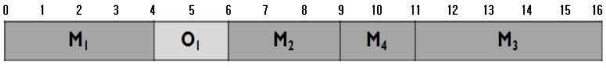
(b) 선택 부분을 지연시킨 경우
(그림 6) 선택 부분 지연에 따른 비교

DOP 알고리즘에서 <표 1>의 태스크의 선택 부분을 제거하는 과정을 설명하면 다음과 같다. <표 1>의 태스크 T_1, T_2, T_3는 도착시간이 모두 0이므로 시간 t=0에서 마감시간에 따라 레디 큐에 정렬되어 있게 된다. 먼저 $M_1 + O_1 + 0 = 7$ 이고, $D_1 = 7$ 이므로 일단 O_1는 남아있게 된다. 다음으로 $M_1 + M_2 + O_1 + O_2 + 0 = 11$이고, $D_2 = 12$ 이므로 O_1, O_2 도 제거하지 않는다. 마지막으로 $M_1 + M_2 + M_3 + O_1 + O_2 + O_3 + 0 = 17$ 이고, $D_3 = 16$ 이 된다. 즉, 남아있는 선택 실행 부분에서 1 단위 시간만큼의 1선택 실행 부분을 제거해야 스케줄링 가능하다는 것을 의미한다. 이 1 단위시간을 마감시간이 늦은 태스크에서 제거해도 현재 상황에서는 최소에러를 만족한다. 하지만, 새로운 태스크가 발생할 경우에 마감시간이 빠른 선택 실행 부분을 앞에서 실행하게 되어 이후에 발생할 태스크를 처리하지 못하게 되는 경우가 생길 수 있다. DOP에서는 이 1 단위시간을 마감시간이 빠른 태스크 T_1에서 제거함으로써 최대 선택 실행 시간을 보장하면서 선택 실행 부분을 마감시간이 늦은 태스크에 할당하여 여유시간을 보다 많이 확보하게 된다. 결과적으로 시간 t=8에서 새로운 태스크 T_4가 발생했을 때 실행이 지연된 T_2와 T_3의 남아있는 선택 실행 부분을 포기함으로써 T_4의 필수 실행 부분을 처리할 수 있게 된다. DOP는 선택 실행 부분의 실행을 지연시킴으로써 이후에 발생할 태스크에 대한 스케줄링 가능성을 높일 수 있게 된다.

<표 1>의 태스크 집합을 DOP로 스케줄링 한 결과는 (그림 7)과 같다. (그림 7)은 새로운 태스크(T_4)가 발생 하지 않은 경우와 발생한 경우에 DOP에서 어떻게 처리되는지를 보여주고 있다. 발생하지 않은 경우는 실행 가능한 선택 실행 부분을 최대로 실행하게 되고, 새로운 태스크가 발생한 경우 지연된 선택 실행 부분을 포기하면서 새로운 태스크의 필수 실행 부분을 처리하는 것을 보여준다.



(a) T_4가 발생하지 않은 경우



(b) T_4가 발생한 경우
(그림 7) 새로운 태스크 발생에 따른 비교 예

MF 알고리즘은 필수 실행 부분을 우선적으로 실행하게 되어 이후에 발생한 태스크(T_4)에 대한 스케줄링 성공률을 높여주었지만, 필수 실행 부분을 우선 실행하게 되어 실행 가능한 선택 태스크의 마감시간이 지나 실행하지 못하여 전체적인 에러가 높아지게 되었다. 만약, 새로운 태스크가 발생하지 않은 경우 NORA 알고리즘과 DOP 알고리즘과 비교하면 같은 필수 실행 부분의 처리율을 보이지만 총 에러가 커지는 것을 알 수 있다.

NORA 알고리즘은 최소 에러를 보장하지만, 선택 실행 부분이 앞에서 실행됨으로써 새로운 태스크 T_4가 발생하는 시점에 T_4의 필수 실행 부분을 예약 리스트에 할당 할 수 없어서 스케줄링하지 못하게 된다. 반면, MF와 DOP는 새로운 태스크를 모두 처리함으로써 NORA 알고리즘에 비해 좋은 필수 실행 부분의 스케줄링 성공률을 보인다.

DOP는 마감시간이 빠른 선택 실행 부분을 적절하게 제거하여 선택 실행 부분의 실행을 지연해 줌으로써 T_4가 발생하기 전의 태스크의 최소 에러를 만족 시키면서 새로운 태스크에 유연하게 대처하는 것을 알 수 있다.

4. 성능 평가

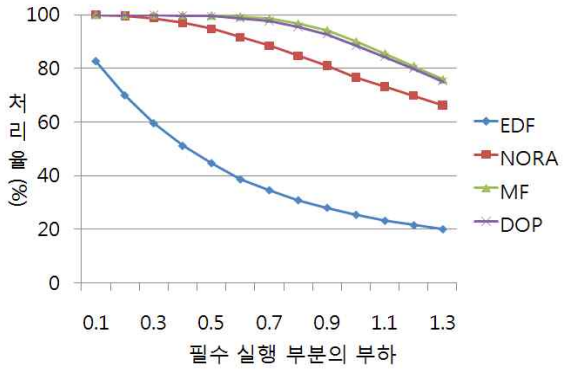
제안 알고리즘의 성능을 평가하기 위한 비교 알고리즘으로 EDF, NORA, MF를 사용했으며, 성능 평가는 필수 실행 부분과 선택 실행 부분의 처리율을 기준으로 비교하였다. 처리율은 발생한 전체 태스크의 실행 부분과 실행이 완료된 태스크의 실행 부분의 비율이다. 필수 실행 부분의 처리율과 선택 실행 부분의 처리율을 각각 구하여 성능을 비교하였다.

$$\text{처리율}(\%) = \frac{\sum \text{처리된 태스크의 실행부분}}{\sum \text{발생한 전체 태스크의 실행부분}}$$

필수 실행 부분의 처리율은 실제 스케줄링 성공률과 비교해 별다른 차이를 보이지 않아 실제 성능 평가에서 태스크 성공률로 보아도 무방하고, 선택 실행 부분은 필수 실행 부분이 완료된 태스크만이 선택 실행 부분을 실행 할 수 있게 하였다. 태스크 생성은 부하에 따라 확률적으로 생성하였다. 즉, 태스크의 필수 부분의 부하가 증가하는 것은 보다 많은 태스크가 생성되는 것을 의미하고, 본 논문에서는 어느 정도 과부하 상황을 고려하여 부하에 대한 평가에서 필수 실행 부분의 부하를 최대 1.3까지 실험하였다.

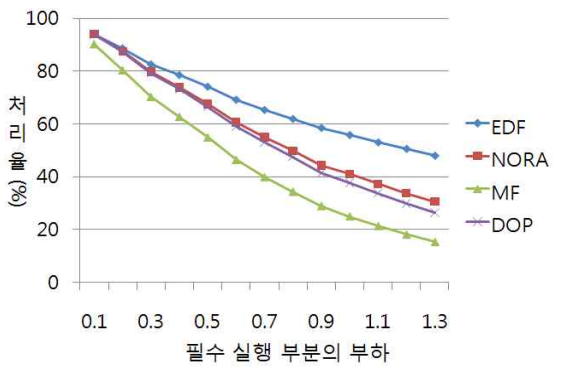
(그림 8)은 필수 실행 부분과 선택 실행 부분의 비율이 1:3인 경우에 처리된 필수 실행 부분의 처리율을 비교한 결과이다. MF 알고리즘은 필수 실행 부분을 우선적으로 처리하기 때문에 가장 높은 필수 실행 부분의 처리율을 보였으며, 사실상 그 이상의 처리율을 보일 수가 없다. DOP 알고리즘이 필수 실행 부분의 처리에서 MF 알고리즘에 근접한 처리율을 보이지만 약간의 차이가 생기게 된다. 이것은 선택 실행 부분의 실행을 최대한 지연시키지만, 최소 에러를

내기위해 일부의 선택 실행 부분이 앞에서 실행됨으로써 차이가 발생하게 된다. NORA의 경우 시스템 부하가 커짐에 따라 필수 실행 부분 처리가 앞의 두 알고리즘에 비해 상당히 떨어지는 모습을 보였다. 이때, 발생한 선택 실행 부분이 많아 세 가지 알고리즘 모두 선택 실행 부분에 대한 처리는 좋은 성능을 내지 못하였다. 필수 실행 부분의 처리에서 MF와 NORA의 경우 최대 약 13%의 차이를 보였고, DOP와 MF는 약 1% 내외의 차이를 보였다. 반면, 필수 실행 부분과 선택 실행 부분의 비율을 3:1로 조절하여 필수 실행 부분의 크기가 더 큰 경우에는 필수 실행 부분의 처리에 있어서 3가지 알고리즘이 별다른 차이를 보이지 않았다.



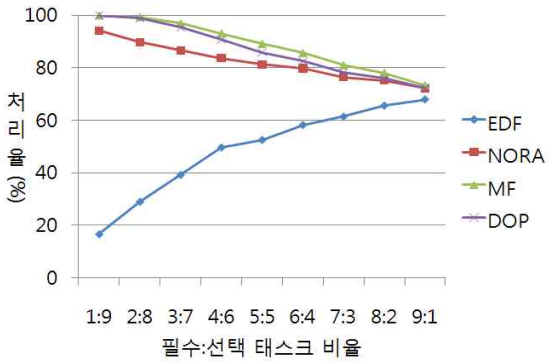
(그림 8) 필수 실행 부분 처리율

(그림 9)는 필수 실행 부분과 선택 실행 부분의 비율이 3:1인 경우, 선택 실행 부분의 처리율을 비교한 것이다. NORA의 경우 가장 많은 선택 실행 부분을 처리하여 적은 에러가 발생하였고, DOP가 NORA에 근접한 선택 실행 부분의 처리를 보였다. DOP가 NORA에 비해 부하가 높아짐에 따라 선택 실행 부분의 처리율이 낮아지는 것은 필수 실행 부분의 수행을 위해 포기된 선택 실행 부분이다. MF의 경우 부하가 높아질수록 선택 실행 부분의 차이가 앞의 두 알고리즘에 비해 점점 커지게 된다. 부하가 높아짐에 따라 NORA와 MF가 약 16%이상의 차이를 보이는 것에 비해, NORA와 DOP는 부하가 큰 경우 약 3%정도의 차이를 보였다. 하지만, 필수 실행 부분과 선택 실행 부분의 비율이 1:3인 경우는 선택 실행 부분의 처리에서 큰 차이를 보이지 않았다.

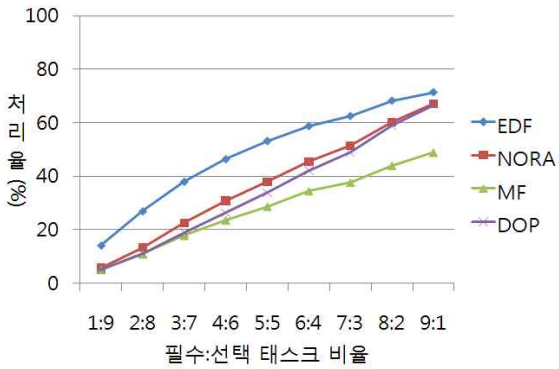


(그림 9) 선택 실행 부분 처리율

(그림 8)과 (그림 9)에서 보여준 결과로 필수 실행 부분과 선택 실행 부분의 비율에서 선택 실행 부분의 비율이 큰 경우는 (그림 8)에서 보는 것처럼 필수 실행 부분의 처리에서 DOP 알고리즘이 가장 뛰어난 필수 실행 부분의 처리율을 가지는 MF 알고리즘에 근접한 성능을 보였으며 NORA 알고리즘은 MF, DOP 알고리즘에 비해 좋은 성능을 내지 못했다. 선택 실행 부분의 처리에 있어서는 NORA, MF, DOP의 3가지 알고리즘이 별다른 차이를 보이지 않았다. 반대로, 필수 실행 부분의 비율이 큰 경우는 필수 실행 부분의 처리는 별다른 차이를 보이지 않은데 비해 (그림 9)에서 보는 것처럼 선택 실행 부분의 처리에서 DOP 알고리즘이 가장 뛰어난 NORA 알고리즘에 근접한 성능을 보였으며, NORA와 DOP 알고리즘에 비해 MF 알고리즘의 선택 실행 부분에 대한 처리율이 떨어지는 것을 알 수 있다. 본 논문과 유사한 DOT의 경우 DOP 알고리즘과 비교해 필수, 선택 부분의 처리율은 1% 정도의 차이를 보인다. 처리하는 태스크에 대한 선점 비율은 DOP의 경우 최소 4%, 최대 11%인 반면, DOT는 최소 23%, 최대 62%의 태스크 당 선점 비율을 가지게 되고, DOT는 DOP에 비해 3-7배 정도의 더 많은 선점 회수를 가진다.



(그림 10) 비율에 따른 필수 실행 부분 처리율



(그림 11) 비율에 따른 선택 실행 부분 처리율

(그림 10)과 (그림 11)에서 보듯이 각 태스크의 비율에 대한 차이가 커짐에 따라 처리율이 보다 많은 차이를 내게 된다. 필수 실행 부분의 비율이 높아지고 선택 실행 부분의 비율이 그만큼 줄어들게 되면, NORA, MF, DOP 알고리즘

의 필수 실행 부분의 처리율이 같아지는 것에 비해 선택 실행 부분의 처리율은 DOP 알고리즘은 가장 뛰어난 NORA 알고리즘의 처리율에 점점 근접한 성능을 보이게 되고, DOP와 NORA에 비해 MF 알고리즘의 성능이 점점 더 크게 떨어져서 처리율의 차이가 보다 확실하게 나타난다. 반대로, 필수 실행 부분의 비율이 줄어들고 선택 실행 부분의 비율이 커지게 되면, NORA, MF, DOP 세 알고리즘의 선택 실행 부분의 처리율이 점점 같아지게 되고, 필수 실행 부분의 처리는 DOP 알고리즘이 가장 뛰어난 MF 알고리즘의 성능에 보다 근접한 성능을 보이고, DOP와 MF 알고리즘에 비해 NORA의 성능이 다소 떨어지게 된다.

5. 결 론

온라인상에서는 이후에 발생 할 태스크를 예측 할 수 없는 비주기적 태스크의 특성으로 인해 최적의 알고리즘이 존재할 수 없다. 더욱이, 실시간 시스템에서 불확정 태스크의 경우는 특정 비율의 태스크만 생성 되는 것이 아니라 다양한 비율의 태스크와 그 태스크의 부하가 낮거나 높아지는 시점을 예측 할 수 없기 때문에 상황에 따라 유연하게 대처할 수 있는 알고리즘이 필요하다.

NORA 알고리즘은 최소 에러를 보장하지만 이후에 발생하는 새로운 태스크에 대한 스케줄링 가능성이 낮아진다. 반면, MF 알고리즘은 새로운 태스크에 대해 높은 스케줄링 가능성을 가지지만 최대 에러 또한 높아지게 되는 문제를 가지고 있다. 두 가지 알고리즘은 특정 태스크 상황 하에 좋은 성능을 내는 것이 증명되었지만, 그 성능이 태스크의 비율에 따라 달라지는 것을 보임으로써 태스크 비율에 많은 영향을 받는다는 것을 알 수 있다.

본 논문에서 제시한 DOP 알고리즘은 현재 태스크 집합의 에러를 최소화 하면서 이후에 발생할 태스크에 대한 필수 실행 부분의 스케줄링 가능성을 높일 수 있는 알고리즘으로서, 태스크의 필수 실행 부분과 선택 실행 부분의 비율에 관계없이 좋은 성능을 가진다. 선택 실행 부분의 비율이 낮은 경우 필수 실행 부분의 처리율은 큰 차이가 나지 않으며 이러한 상황에서는 선택 실행 부분의 처리가 중요하게 된다. DOP는 이러한 경우에 낮은 에러를 내는 NORA 알고리즘과 비슷한 에러를 가진다. 반면, 필수 태스크의 비율이 낮은 경우에는 필수 실행 부분의 처리가 중요하게 되고, DOP는 뛰어난 필수 실행 부분의 처리율을 가진 MF 알고리즘과 근접한 필수 실행 부분 처리율을 보여주었다.

향후 연구로 태스크의 제거가 아닌 이후의 태스크에 대해 보다 향상된 처리율을 내기 위해 선택 태스크 뿐만 아니라 필수 태스크 또한 적절하게 조절을 하는 방법에 대한 연구가 필요하다. 실시간 시스템이 과거의 단순한 기능에서 보다 복잡한 기능을 수행할 수 있도록 요구되는 추세에 따라 단일프로세서 뿐만 아니라 다중프로세서에 적용할 수 있는 방법의 연구가 필요하다.

참 고 문 헌

[1] J. Liu, K. Lin, W. Shih, A. Yu, J. Chung, and W. Zhao, “Algorithms for scheduling imprecise computation,” Computer, Vol.24, No.5, May, 1991.

[2] Shih, W.-K., Liu, J.W.S., Chung, J.-Y., "Fast algorithms for scheduling imprecise computations", In proceedings of the Real Time Systems Symposium, IEEE Computer Society Press, 1989.

[3] Shih, W.-K., Liu, J.W.S., "On-line scheduling of imprecise computations to minimize error", Real-Time Systems Symposium, 1992.

[4] Jai-Hoon Kim, Kihyun Song, Kyunghee Choi, Gihyun Jung, SeunHun Jung, "Performance evaluation of on-line scheduling algorithms for imprecise computation", Real-Time Computing Systems and Applications, 1998.

[5] Cho, S., Lee, S., Ahn, S., and Lin, K., "Efficient Real-Time Scheduling Algorithms for Multiprocessor Systems," IEICE Trans, Commun., Vol.E85-B, No.12, pp.2859-2867, 2002.

[6] Giorgio C. Buttazzo, "Hard real-time computing systems", 2nd Ed., Springer, 2005.

[7] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment", Journal of the ACM, Vol.20, No.1, pp.46-61, 1973.

[8] Liu, J. W., Real-Time Systems, p.70, Prentice Hall, 2000.

[9] Jia-Ming Chen, Wan-Chen Lu, Wei-Kuan Shih and Ming-Chung Tang, "Imprecise Computations with Deferred Optional Tasks", Journal of Information Science and Engineering 25, 185-200, 2009.

[10] Radhakrishna Naik, R. R. Manthalkar, "Modified IUF Scheduling Algorithm for the Real Time Systems," ICETET-10,IEEE third International Conference on Emerging Trends in Engineering & Technology, pp.712-716, 2010.

[11] Damir Poles, Leo Budin, "Imprecise Computation Model, Synchronous Periodic Real-time Task Sets and Total Weighted Error", Journal of Computing and Information Technology, Vol.18, No.4, 2010.

최 환 필

e-mail : rdzmoon@gmail.com

2009년 강원대학교 컴퓨터정보통신공학부 (학사)

2009년~현 재 강원대학교 컴퓨터정보통신공학부 석사과정

관심분야 : 실시간 시스템, 운영체제, 알고리즘

김 용 석

e-mail : yskim@kangwon.ac.kr

1984년 서울대학교 해양학과(학사)

1986년 KAIST 전기및전자공학과(공학석사)

1989년 KAIST 전기및전자공학과(공학박사)

1990년~1995년 한국생산기술연구원 및 전자부품연구소 선임연구원

1995년~현 재 강원대학교 컴퓨터학부 교수

관심분야 : 운영체제, 실시간 시스템, 임베디드 시스템