멀티미디어 시스템에서 QoS 지원을 위한 불확정계산 기반의 스케줄링

김 태 수, 김 용 석
강원대학교 컴퓨터정보통신공학과
e-mail : etskim@*mail.kangwon.ac.kr*, yskim@*kangon.ac.kr*

Imprecise Computation based Scheduling for QoS Support in Multimedia Systems

Tae-Su Kim, Yong-Seok Kim
Dept of Electrical and Computer Engineering
Kangwon National University

Abstract

A task in imprecise computation consists of mandatory part and optional part. The optional part can be executed partially and the quality of service is measured by the amount of the execution. Many paper showed that multimedia systems are good applications of imprecise computation. It is important to guarantee QoS which is a critical factor in multimedia systems. Previous works didn't consider QoS and processor slack were assigned randomly to tasks.

This paper presented a systemic slack assignment method according to QoS levels of tasks. A simulation result showed that our method can be a good choice for multimedia systems with QoS requirement.

I. 서론

불확정계산(Imprecise Computation)은 태스크를 필수부분(Mandatory part)과 선택부분(Optional part)으로 나누어 실행하는 것이다. 따라서 사용되는 태스크는 결과의 정확도를 판단하기위한 필수부분과 실행결과의 질에 영향을 미치는 선택부분으로 나눌 수 있어야 한다. 이러한 특징의 태스크를 사용하는 대표적인 예는 멀티미디어 시스템이다.

QoS(Quality of Service)는 멀티미디어 시스템이 가져야할 핵심적인 요소이다. 하지만 지금까지 제안된 불확정계산 기반의 멀티미디어 태스크 스케줄링 기법은 시스템 활용도와 필수부분의 실행에 그 목적이 맞추어져 있을 뿐 QoS를 고려하지 않고 있다. 따라서 본 논문에서는 QoS의 수준에 맞는 여유시간 분배 기법을 통해 멀티미디어 시스템에서 QoS를 지원하는 방법을 제안한다.

II. 관련 연구

2.1 불확정계산 실시간 스케줄링

불확정계산에 대한 실시간 스케줄링은 필수부분의 데드라인을 보장하고 여유시간에 선택부분을 효율적으로 실행하는 것이 목적이다. 초창기부터 광범위하게 연구된 오프라인 스케줄링은 몇 개의 최적화된 알고리즘이 존재한다.[1, 2, 3]. 반면, 온라인 스케줄링은 최적화된 알고리즘은 없지만 다음 두 가지의 정책이 존재한다. 첫 번째는 필수부분의 우선순위를 선택부분보다 높게 두고 실행하는 방법이며[4, 5], 두 번째는 시스템의 여유시간을 선택부분을 위해 할당하는 방법[6]이다.

2.2 SS-OP (Slack Stealer for Optional Part)

SS-OP 알고리즘은 단일 프로세서 시스템에서 온라인 선점 스케줄링을 사용하여 데드라인이 존재하는 불확정계산 작업을 스케줄링하는 알고리즘이다. 전통적인 불확정계산 모델에 마무리 부분(Windup part)을 추

가한 모델을 사용하며, 마무리부분은 해당 작업이 어떠한 상황이든 해당 작업을 마무리하고 다음 실행시 이전 작업이주는 영향을 제거한다.[7]

SS-OP는 기본적으로 EDF(Earliest Deadline First)를 이용하여 태스크의 필수부분을 실행한다. 그리고 태스크의 선택부분은 데드라인과 선택부분 활용률을 이용하여 여유시간을 할당하여 실행한다. 여유시간 할당의 우선순위는 데드라인이 짧은 태스크가 가장 높다. 태스크가 준비되면 자신의 실행 영역에서 낮은 우선순위의 태스크가 가진 여유시간을 빼앗아오고 높은 우선순위의 태스크의 데드라인 이후에서 여유시간을 시스템으로부터 할당받는다. 태스크가 종료할 때 남긴 여유시간 역시 다른 태스크의 선택부분 실행을 위해 전달된다. 이때 역시 여유시간을 전달받는 태스크는 EDF적 우선순위가 가장 높은 태스크이다.

위에서 살펴보았듯이 SS-OP는 여유시간 분배에 있어 현재 가장 데드라인이 짧은 태스크가 최고의 우선순위를 갖게 된다. 특정 시점에서 어느 태스크가 가장 큰 우선순위를 갖고 있는지는 알 수 없으므로, 각 클라이언트의 우선순위가 따로 정해진 멀티미디어 시스템에서 SS-OP는 적합하지 않다.

III. QSS-OP (QoS SS-OP)

3.1 가정과 용어

QSS-OP는 주기와 데드라인이 같은 주기적인 불확정계산 태스크를 대상으로 스케줄링 하는 알고리즘이며, 단일 프로세서 시스템에서 사용되는 선점형 온라인 스케줄링기법이다. 불확정계산 모델 및 EDF 기반의 필수부분 스케줄링방법은 SS-OP와 동일하다.

QSS-OP는 태스크 i를 위해 다음과 같은 기호들을 사용한다.

표 1. 용어 정리

기호	설명
Mi	필수부분 실행 시 소요되는 최악의 시간
Wi	마무리 부분 실행 시 소요되는 최악의 시간
Ti	현 태스크의 주기 (상대적 데드라인)
Di	현 태스크의 실제 데드라인
Ri	실제 실행하기 위해 i에게 할당된 시간
Si	i가 가지고 있는 여유시간
Task State(i)	현 태스크의 상태로 다음 4가지 경우 중 하나. MANDATORY, OPTIONAL, WINDUP, END
$QoS(i)$	태스크의 QoS 수준

태스크가 사용할 수 있는 여유시간의 양을 알기위해서는 선택부분 활용률을 알아야한다. 선택부분 활용률은 각 태스크의 필수부분 활용률 Ui 와 전체 필수부분 활용률 Ue를 이용하여 구한다. 해당 식은 다음과 같다.

$$Ui = \frac{Mi + Wi}{Ti}, \quad Ue = \sum_{i=1}^{n} Ui, \quad Uo = 1 - Ue$$

QSS-OP에는 SlackOwner와 ASL(Alloted Slack Line), SystemSlack이라는 새로운 개념을 사용한다. 우선 SlackOwner는 현재 실행되고 있는 태스크 중 QoS가 가장 높은 태스크를 나타낸다. SlackOwner의 데드라인은 실행중인 태스크 중 가장 짧을 수도 있고 그렇지 않을 수 도 있다.

ASL은 현재까지 SlackOwner에게 할당된 여유시간의 경계선을 말한다. 다음 SlackOwner가 어디서부터의 여유시간을 할당 받아야 하는지 알려주며 여유시간의 중복 할당을 막는다.

SystemSlack은 현재 SlackOwner가 존재하지 않을 때 앞에서 전달된 여유시간을 시스템에서 유지하기 위해 사용된다.

3.2 알고리즘

QSS-OP는 항상 SlackOwner의 값을 유지하면서 새로운 SlackOwner가 결정되면 SlackOwner에게 여유시간을 할당한다. 그리고 이외의 시간에는 자신에게 할당된 시간을 소모 한다.

```
1. 태스크가 준비시
   Ri = Mi,  TaskState(i) = MANDATORY
   if QoS(i) > QoS(so) then
     이전 SlackOwner로부터 여유시간을 가져온다.(3.4절)
     Si = Si + SystemSlack, SystemSlack = 0,
     Sso = 0
     SlackOwner = i, 그림2.
   else
     Si = 0
2. 필수부분 완료시
   Ri = Ri + Si, Si = 0, TaskState(i) = OPTIONAL
3. 선택부분 종료시 또는
     태스크에게 할당된 시간이 없을 경우 ( Ri = 0)
   Ri = Ri + Wi,  TaskState(i) = WINDUP
4. 마무리부분 완료시
   TaskState(i) = END
   if SlackOwner == i then
     if 새로운 SlackOwner 발견 then
       그림2. Sso = Sso + Ri
     else
       SystemSlack = Ri
   else
     Sso = Sso + Ri
5. 실행중인 태스크가 없을 경우
   SystemSlack = SystemSlack − 경과시간
```

그림 1. 태스크 상태 별 QSS-OP 알고리즘

그림 1은 QSS-OP의 알고리즘을 작업의 진행 여부에 따라 설명한 것이다. 태스크가 처음 준비되면 필수부분을 위한 시간을 할당받는다. 그리고 태스크의 QoS가 SlackOwner의 QoS보다 높으면 여유시간을 할당하고, ASL을 조정한다. 이후 필수부분과 선택부분을 실행하면서 자신이 가지고 있는 사용 가능시간을 소모한다.

SlackOwner가 종료했을 경우 다음 SlackOwner를 찾는다. 다음 SlackOwner를 찾으면 해당 태스크에게 남은 시간을 전달하고, ASL을 조정한다. 만일 찾지 못하면 SystemSlack에 남은 시간을 보존하여 여유시간의 유실을 막는다. SlackOwner가 종료하는 것이 아니면 기존에 존재하는 SlackOwner에게 남은 시간을 전달한다. 시스템이 실행되는 태스크 없이 유지되면 해당 시간만큼 SystemSlack을 조절한다.

3.4 여유시간의 할당과 ASL의 조정

여유시간을 할당 받고 ASL을 조정하는 시기는 새로운 SlackOwner가 준비되었을 경우나 SlackOwner가 끝났을 때이다. 여유시간을 할당 받는 방법은 ASL을 기준으로 ASL 이전의 여유시간은 기존의 SlackOwner로부터 빼앗아오고, ASL이후의 여유시간은 시스템으로부터 할당 받는다. 그리고 여유시간의 중복 할당을 막기 위해 ASL을 조정한다.

ASL이전의 여유시간 할당을 살펴보면 태스크 준비시에는 이전 태스크의 상황에 맞게 태스크가 가지고 있는 여유시간(Ri, Si, 또는 모두)을 모두 가져오고, SlackOwner가 끝났을 경우에는 이전 SlackOwner가 없으므로 아무것도 받아 오지 않는다.

1. ASL 이후의 여유시간 할당

$$Sso = \begin{cases} Sso + (Dso - ASL) * Uo & \text{if } ASL < Dso \text{ then} \\ Sso - (ASL - Dso) * Uo & \text{else} \end{cases}$$

2. ASL 조정

$$ASL = \begin{cases} Dso - \frac{Sso}{Uo}, Sso = 0 & \text{if } Sso < 0 \text{ then} \\ Dso & \text{else} \end{cases}$$

그림 2. ASL 이후의 여유시간 할당과 ASL의 조정

그림 2는 자신의 데드라인과 ASL을 이용하여 여유시간을 할당받고, 자신이 할당 받은 만큼 ASL을 조정하는 방법을 보여주는 그림이다. 태스크가 준비되었을 때는 ASL과 데드라인의 크기를 비교하여 ASL이 데드라인보다 클 경우 자신이 초과하여 받은 여유시간을 돌려주고 그에 맞게 ASL을 조정한다. SlackOwner 종료 후 새로 선택된 SlackOwner는 항상 ASL보다 데드라인이 길다.

Ⅳ. 성능 측정

5.1 SS-OP와 QSS-OP의 비교

시뮬레이션을 위한 테스트 데이터는 주기, 필수부분, 마무리부분이 각각 [20, 60], [10, 40], [1, 9]의 값들 사이에서 무작위하게 선택되었으며 선택부분은 필수부분과 연관하여 $[2*Mi-15, 2*Mi+5]$중 무작위로 선택되었다.

하나의 태스크 집합에는 각기 다른 QoS 수준을 가진 3개의 태스크가 존재한다. 시뮬레이션을 평균 필수활용률은 0.62 1000단위시간, 50회 반복하여 실행하였다.

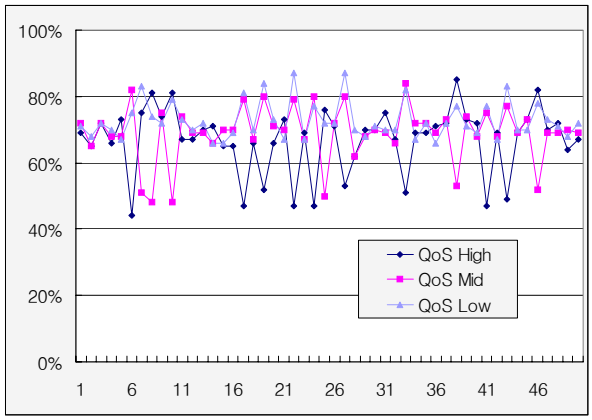

그림 3. QoS 수준별 SS-OP의 선택부분 성공률

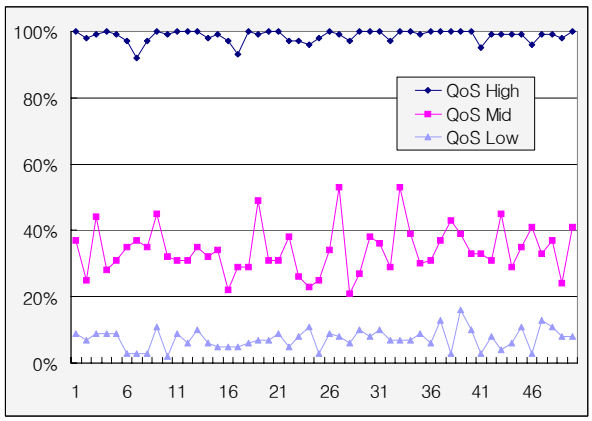

그림 4. QoS 수준별 QSS-OP의 선택부분 성공률

그림 3과 4는 3개의 태스크가 자신이 원하는 선택부분 실행양중 실제 실행한 양을 백분율로 보여주는 그림이다. 그림 3의 그래프를 보면 각 태스크가 50회를 실행하면서 QoS의 수준에 관계없이 매번 무작위한

우선순위로 실행된 것을 볼 수 있다. 이는 SS-OP가 여유시간 할당에 있어서 QoS를 고려하지 않는다는 것을 보여준다. 그림 4는 각 태스크가 자신의 QoS 레벨에 맞추어 선택부분을 실행한 것을 보여준다. 이는 QSS-OP가 QoS 수준이 높은 태스크에게 더 많은 여유시간을 할당한 것을 뜻한다.

5.2 필수 활용률과 성능

이전 실험에서 필수 활용률은 평균 0.62였으므로, 이번에는 각기 다른 필수 활용률에서 어떻게 작동할 것인가에 대해 실험하였다. 이전 시뮬레이션을 다른 필수 활용률로 실행 하였다. 필수 활용률의 범위는 0.56에서 0.96이다.

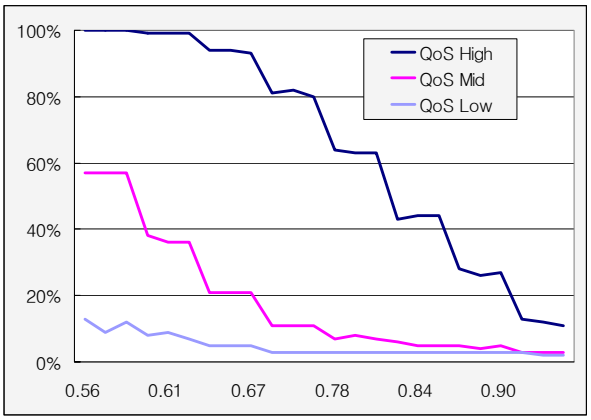

그림 5. 필수 활용률에 따른 선택부분 성공률

그림 5는 필수 활용률의 변화에 따른 선택부분 성공률의 변화를 나타내는 그래프이다. 필수 활용률의 변화와 관계없이 QoS의 선택부분 성공률이 QoS수준에 맞추어 나타났다. 이는QSS-OP가 필수 활용률에 관계없이 QoS에 수준을 고려하여 여유시간을 할당한 것을 보여준다.

VI. 결론

불확정계산을 이용한 멀티미디어 시스템에서는 QoS를 위한 태스크 스케줄링이 필요하다. QoS를 지원하기 위해서는 필수부분의 실행과 QoS를 고려한 차등적인 여유시간 분배가 필요하다. 하지만 기존의 스케줄링 기법에는 QoS를 고려한 여유시간 분배 기법이 없다.

본 논문에서는 태스크의 QoS의 지원을 위해 여유시간 분배 알고리즘을 제시하였다. QSS-OP에서 모든 태스크는 EDF 정책에 따라 스케줄링이 이루어지며, 시스템의 여유시간은 각 태스크 별로 QoS의 수준에 따라 적절히 분배된다. 그 결과 기존의 스케줄링 기법보다 평등한 QoS를 보장할 수 있게 되었다.

참고문헌

[1] J.W.S.Liu and W. K. Shih. Algorithms for Scheduling Imprecise Computations with timing Constraints to Minimize Maximum Error. IEEE Trans. Comput., 1995.

[2] W.-K. Shih, J.W.S.Liu, and J.-Y.Chung. Algorithms for Scheduling Imprecise Computations with Timing Constraints. SIAM J. Comput., June 1991

[3] H. Aydin, P.Mejia-Alvarez, R. Melhem, and D. Mosse. Optimal Reward-Based Scheduling of Periodic Real-Time Tasks. In Proceedings of the 20th EEEE Real-Time Systems Symposium, Dec. 1999.

[4] J.-Y. Chung, J.W.S.Liu, and K,-J Lin. Scheduling Periodic Jobs That Allow Imprecise Results. IEEE Trans. Comput., Sept. 1990.

[5] S. Baruah and M. Hickey. Competitive On-Line Scheduling of Imprecise Computations. IEE trans. Comput., Sept. 1998.

[6] W.-K.Shih, J.W.S.Liu. On-Line Scheduling of Imprecise Computations to Minimize Error. SIAM J. Comput., 1996.

[7] Hidenori Kobayashi and Nobuyuki Yamasaki, RT-Frontier : A Real-Time Operation System for Practical Imprecise Computation, 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2004.