조합형 실시간 스케줄링의 양자화 문제

유시환, 유혁
고려대학교 정보통신대학
{shyoo,hxy}@os.korea.ac.kr

Quantization Analysis in Compositional Real-time Scheduling

Seehwan Yoo, Chuck Yoo
Korea University

요 약

조합형 실시간 스케줄링은 계층적으로 구성된 실시간 시스템에 대해 실시간성을 보장할 수 있는 방법을 제공한다. 조합형 스케줄링 이론을 통해 여러 개의 실시간 태스크를 하나의 실시간 태스크로 묶어 스케줄링 할 수 있으며, 실시간 보장을 위해 필요한 CPU 요구량을 계산하였다. 하지만, 양자화에 대한 고려가 없어, 틱-기반 스케줄링 시스템에서 정확한 CPU 요구량을 계산할 수 없다. 따라서, 본 연구에서는 양자화에 따른 CPU 할당량의 영향을 정량적으로 보여준다.

1. 서 론

실시간 스케줄링에 대한 필요성이 기존의 단순한 임베디드 시스템에서 벗어나 보다 복잡한 시스템으로 이전되면서, 복잡한 시스템에 대한 실시간성을 보장해 줄 수 있는 이론이 등장하게 되었다. 계층형 실시간 스케줄러는 이처럼 복잡한 시스템에서 실시간 스케줄링을 제공할 수 있는 방법을 제공한다. 계층형 실시간 스케줄러는 실시간 태스크들을 특성에 따라 여러 개의 태스크 그룹으로 분류하고, 태스크 그룹 간의 스케줄러와 태스크 그룹 내의 스케줄러 간의 계층을 두어 모든 태스크들에 대한 실시간 스케줄링을 보장한다.

계층형 실시간 스케줄링 중 조합형 실시간 스케줄링은 태스크 그룹 내부의 스케줄링 정책을 유지하면서도 실시간 속성을 보장하기 위해 필요한 (태스크 그룹 당) CPU 요구량을 계산한다. 즉, 태스크 그룹 내 태스크들의 실시간 보장을 위해 필요한 CPU 요구량을 분석하고, 상위 계층의 태스크 그룹 스케줄러가 보장해 주는 CPU 제공량과 비교하여 계층적으로 구성된 실시간 시스템의 스케줄링 가능성을 분석할 수 있는 기법을 제공한다.

하지만, 조합형 스케줄러를 통해 분석해 낸 CPU 요구량은 틱-기반의 스케줄링 시스템에서 실시간 보장을 위해 사용할 수 없다. 틱-기반의 스케줄링 시스템들은 CPU 사용량을 일정한 시간 단위로 분할하여 스케줄링하게 되는데, 이와 같이 분할된 시간 단위가 CPU 요구량을 정확하게 표시할 수 없는 경우, 불가피하게 오버헤드가 발생하게 된다.

이 논문에서는 스케줄링의 정확도와 관련한 틱-기반의 스케줄링 문제를 양자화 문제로 정의하고, 조합형 스케줄러가 가진 양자화 문제를 분석한다.

논문의 구성은 다음과 같다. 2장에서는 조합형 스케줄링과 관련된 연구들을 소개하며 한계점을 밝힌다. 3장에서는 양자화 문제를 정의하고, 이로 인해 발생하는 전체 오버헤드를 밝히고, 4장에서는 논문의 결론을 도출한다.

2. 관련 연구

2.1 조합형 실시간 스케줄러의 이론적 배경

조합형 실시간 스케줄링 이론[1][2]은 여러 실시간 태스크들을 묶어서 하나의 주기 태스크로 변환함으로써 상위의 스케줄러에서 손쉽게 태스크 그룹의 실시간 실행을 보장해 줄 수 있다. 여러 개의 주기적 태스크를 가지는 태스크 그룹 $G=\{T_i(p_i,e_i)\}$ 은 주기적 조합 태스크 $\Gamma(\Pi,\Theta)$ 로 대표된다. 태스크 그룹간 스케줄러는 조합 태스크 만을 스케줄링 함으로써 스케줄링의 단순함을 유지하고, 태스크 그룹은 내부의 스케줄링 알고리즘을 독자적으로 사용할 수 있다.

조합형 실시간 스케줄링에서는 태스크 그룹에 제공되는 자원 제공량과 태스크 그룹이 실시간 실행을 위해 필요한 자원 요구량을 계산하여 스케줄 가능성을 분석한다.

우선, 자원 요구량은 태스크 그룹 내의 모든 태스크의 실시간성이 보장되기 위해 필요한 최소한의 CPU 요구량이다. 따라서, 자원 요구량은 태스크 그룹 내의 요소들에 의해 결정된다. 자원 요구량은 태스크 그룹 내의 스케줄링 알고리즘과 태스크들의 특성에 따라 결정되는데, 예를 들어, 로컬 스케줄러가 EDF (Earliest Deadline First)인 경우, 임의의 t 시간 동안 필요로

하는 자원의 최소 요구량은 (1)과 같다.

$$dbf_{EDF}(G,t)=\left(\sum\frac{e_i}{p_i}\right)t \quad , \tag{1}$$

이 때, p_i 와 e_i 는 각각 태스크 T_i 의 주기와 실행시간이다. 만약 로컬 스케줄링으로 RM (Rate Monotonic) 을 사용하는 경우, t 시간 동안의 자원 요구량은 다음 (2)와 같이 계산된다.

$$dbf_{RM}(G,t,i)=e_i+\sum_{T_k\in HP_G(i)}\left\lceil\frac{t}{p_k}\right\rceil e_k, \tag{2}$$

자원 제공량은 조합 태스크가 임의의 t 시간동안 실행될 때, 반드시 보장받는 최소한의 자원량이다. 조합 태스크는 주기적인 실행을 하므로, 최악의 경우에 대해 보장받을 수 있는 자원 제공량을 다음 (3)과 같이 계산할 수 있다.

$$sbf_{\Gamma}(t)=\begin{cases} t-(k+1)(\Pi-\Theta) \\ \quad \text{if } t\in[(k+1)\Pi-2\Theta,\ (k+1)\Pi-\Theta], \\ (k-1)\Theta \\ \quad \text{otherwise.}\end{cases} \tag{3}$$

이 때, k는 주기-배수 관계로 정의된 변수이며, t 시간 동안 최대 수행 가능한 주기의 수가 된다. 정확한 periodic multiple의 정의는 [1]를 참조하도록 한다. EDF와 RM의 경우 k는 다음과 같이 정의 된다.

$$K_{RM}(P_{\min},\Gamma(\Pi,\Theta))= \max\{k\in\mathbb{N}\mid(k+1)\Pi-\Theta<P_{\min}\} \tag{4}$$

$$K_{EDF}(P_{\min},\Gamma(\Pi,\Theta))= \max\{k\in\mathbb{N}\mid(k+1)\Pi-\Theta-\frac{k\Theta}{k+2}<P_{\min}\}. \tag{5}$$

스케줄 가능성 검사는 주기-배수 관계를 이용하여, 주어진 자원의 최소값이 태스크 그룹 내부의 모든 실시간 태스크들이 필요로 하는 자원 요구량 이상을 갖게 되는지 확인함으로써 이루어진다. [2]에서 밝힌 스케줄 가능성 검사는 로컬 스케줄링 정책에 따라 다음과 같이 정의 된다.

EDF의 경우, 태스크 집합 G에 대해 주기적 스케줄링 $\Gamma(\Pi,\Theta)$ 를 통해 스케줄링되는 경우, 다음 (6)과 같은 조건에서 스케줄 가능하다.

$$\forall 0<t<LCM_G,\qquad dbf_{EDF}(G,t)\le sbf_{\Gamma}(t), \tag{6}$$

이 때 LCM_G 는 태스크 그룹 G의 모든 태스크들의 주기의 최소공배수이다.

RM의 경우, 태스크 집합 G 에 대해 주기적 스케줄링 $\Gamma(\Pi,\Theta)$ 를 통해 스케줄링되는 경우, (7)과 같은 조건에서 스케줄 가능하다.

$$\forall T_i\in G,\ \exists t_i\in[0,p_i] \quad s.t.\quad dbf_{RM}(G,t_i,i)\le sbf_{\Gamma}(t_i). \tag{7}$$

이를 바탕으로 기존 연구에서는 스케줄 가능한 workload utilization의 한계치와 임의의 태스크 집합에 대해 스케줄 가능하도록 하는 태스크 그룹의 CPU 요구량을 계산하였다.

특히, 태스크 그룹의 CPU 요구량은 계층적 스케줄링 모델에서 상위 계층에 요구사항으로 반영되어 태스크 그룹 단위의 스케줄 가능성이 만족되도록 보장하는 CPU 예약을 가능하도록 한다.

이는, 태스크 단위의 스케줄링을 태스크 그룹 단위에 대해 스케줄링을 적용하면서 발생하는 오버헤드가 포함되어, 실제 태스크들의 CPU 활용율보다 높은 값을 갖게 된다. EDF와 RM의 경우에 대하여 태스크 그룹의 CPU 요구량인 Abstract bound(추상화 임계값)은 다음 (8)과 같이 정의된다.

$$AB_{G,EDF}(k)=\frac{(k+2)U_w}{k+2U_w},\quad k=K_{EDF}(P_{\min},\Gamma(\Pi,\Theta)).$$
$$AB_{G,RM}(k)=\frac{U_w}{\log\left(\frac{2k+2(1-U_w)}{k+2(1-U_w)}\right)},\quad k=K_{RM}(P_{\min},\Gamma(\Pi,\Theta)). \tag{8}$$

구체적인 내용의 증명은 [1]을 참조하도록 한다.

추상화 임계값의 정의로부터 추상화 오버헤드를 다음 (9)와 같이 정의할 수 있다.

$$\Psi(\Pi)=AB(k)-U_w. \tag{9}$$

앞으로는 혼동이 없는 경우에 한해, AB(k)를 $AB_{G,Algo}(k)$와 혼용하여 사용하도록 한다.

본 논문에서는 단순한 주기 실행 모델을 통해 자원 모델의 자원 제공 함수와 요구량 함수를 계산한 [2]에 기반하고 있다. 현재 조합형 스케줄링 이론은 명시적 주기 실행 모델 (Explicit Deadline Periodic)과 같은 다른 자원 모델을 사용하여 효율성을 높일 수 있는 접근을 취하고 있으며[3], 멀티 프로세서 환경 등을 고려한 확장 역시 연구되고 있다[4].

이 외에도 계층적 실시간 스케줄링을 다룬 여러 가지 이론들이 제시되어있다. [5]에서는 자원 사용 시간을 파티셔닝하여, 실시간 스케줄 가능성 검사를 제공하는 기법을 제시하였다. 이러한 파티션 스케줄러는 [6]에서 재귀적인 방식의 자원 파티션을 지원함으로써 일반화되었으며, BD(Bounded-Delay) 파티션 모델을

지원하여 static 파티션 모델보다 효율적인 스케줄링이 가능하다.

파티션 스케줄러와 조합형 스케줄러는 모두 계층적 스케줄러 시스템에서 CPU 할당을 통해 실시간 실행을 보장한다. 하지만, CPU 할당의 문제는 실제 시스템의 틱-기반 스케줄링 모델과 일치하지 않는다. 이 논문에서는 이러한 문제를 다루고 있으며, 보다 자세한 논의는 다음 장에서 하도록 한다.

2.2 이산 시간 시스템에서의 스케줄링과 양자화

이산 시간 이벤트 시스템에서 스케줄링의 양자화와 관련된 문제는 스케줄링 시스템에서 오래 전부터 다루어왔던 문제이다. 특히, 실시간 시스템의 주기적 특성을 보완하기 위해 제시된 [7], [8] 등의 논문에서는 비율 공평 스케줄러를 제안하고 있으며, 이를 이산 시간 시스템에서 구현할 수 있는 PD, PD^2 등의 알고리즘들이 [9], [10] 에서 밝혀져 있다. 하지만, 조합형 스케줄러나 파티션 스케줄러와 같이 계층형 스케줄러에 대해 양자화 문제가 분석된 경우는 없으며, 보다 구체적으로 PD나 PD^2 알고리즘의 경우, 가상 임계시간(pseudo deadline)을 정의하는데, 태스크 그룹에 대해 가상 임계 시간을 얻어내기 힘들기 때문에 실제로 계층형 시스템에서 이러한 알고리즘을 그대로 사용하기는 힘들다.

양자화와 관련하여 틱-간격과 관련한 오버헤드는 유사한 문제에 대하여 다른 관점을 보여준다. 운영체제가 스케줄링을 하는 기준이 되는 타이머 인터럽트는 높은 정밀도를 가질수록 큰 오버헤드를 갖는다. 즉, 타이머 인터럽트가 자주 들어오는 시스템은 보다 자주 스케줄링을 실행할 기회를 가지며, 따라서 세밀하고 정교한 스케줄링이 가능해 진다. 하지만, 타이머 인터럽트를 아주 작은 수준으로 낮추는 것은 매우 힘든 작업이다. 운영체제가 인터럽트 처리와 스케줄링에 따른 오버헤드를 가지기 때문이다. 특히, 최근의 다중 프로세스 주소공간을 지원하는 운영체제들은 태스크 스위칭에 있어서 매우 큰 오버헤드를 갖게 된다. 따라서, 현실적으로 대개의 운영체제들에서는 1ms 이나 10ms 정도의 정밀도를 갖는 타이머 인터럽트 처리를 통해 스케줄링의 정밀도를 보장하고 있다.

이러한 타이머 정밀도와 태스크 스위칭으로 인한 오버헤드는 처리량과 스케줄링 정밀도를 트레이드-오프 관계로 보고 적절한 정밀도와 처리량을 유지할 수 있는 타이머 인터럽트 간격을 찾도록 한다.

반면, 본 논문에서 제기하는 양자화 문제는 CPU 할당량과 밀접한 관련이 있다. 본 논문에서는 CPU 할당량이 스케줄링 주기에 의해 영향을 받음에 착안하여, CPU 할당량과 스케줄링의 정밀도를 트레이드-오프 관계에서 해석한다. 즉, CPU 할당량이 틱 단위의 실행시간으로 정확하게 표현되지 못하는 경우에 대한 문제를 다루고 있다.

하지만, 두 가지 문제는 밀접한 관련이 있다. 실제 시스템에서는 스케줄링 정밀도를 높이기 위하여, 틱-간격을 줄이는 경우, 절대적인 주기당 틱의 수가 늘어나게 되어, 결국 비슷한 효과를 갖게 된다. 이 경우에는 처리량을 희생하지만, 높은 스케줄링 정밀도를 갖게 된다. 이와 관련한 보다 복잡한 분석은 이 논문의 범위를 벗어난다.

3. 양자화 문제와 CPU 할당량 변화

3.1 양자화 문제의 정의

조합형 스케줄러를 통해 제시된 추상화 임계값은 계층적 실시간 스케줄링을 위해 매우 손쉽게 사용될 수 있다. 즉, 추상화 임계값은 주어진 태스크 집합을 스케줄링 할 수 있는 CPU 할당량을 계산해 주므로, CPU 자원 예약과 같은 비교적 간단한 스케줄링 방식을 통해 실시간성을 보장해 줄 수 있다.

하지만, 필요로 하는 CPU 할당량이 스케줄러의 파라미터로 표현 가능하지 않은 경우, 추가적인 오버헤드가 발생하며, 양자와 오버헤드는 그 중 대표적인 오버헤드이다. 즉, CPU 할당량이 0과 1 사이의 실수로 주어지는데 반해, 태스크 그룹의 (주기, 실행 시간)은 일정한 시간 단위로 변환되어 사용되어야만 한다. 따라서, 스케줄러가 필요로 하는 (주기, 실행시간)은 CPU 할당량인 추상화 임계값으로 표현되어야 하는데, 주기와 실행시간은 모두 정수형의 시간 단위가 되므로, 항상 추상화 임계값을 정확하게 얻어낼 수는 없다.

예를 들어, 태스크 그룹의 주기를 1로 설정한 경우, 추상화 임계값이 10~20%에 머문다고 하더라도, 실제로는 실행 시간을 0.1이나 0.2가 아닌 1로 제공해야 하며, 이는 100%의 CPU가 예약 되어야함을 의미한다.

실제로 이산시간 이벤트 시스템에서는 시간 단위를 타이머 인터럽트 혹은 틱 이벤트를 통해 정의하며, 틱 값은 양의 정수만을 가진다.

3.2 양자화 오버헤드

추상화 임계값은 태스크 그룹의 실시간 실행을 보장하기 위하여 필요한 최소 CPU 할당량이므로, 실제 CPU 할당을 위해서는 추상화 임계값 이상의 CPU 할당이 필요하다. 양자화 문제를 고려하지 않은 경우 태스크 그룹의 스케줄링을 결정하는 실행시간은 다음 (10)과 같이 결정된다.

$$\Theta = AB \cdot \Pi \tag{10}$$

하지만, 양자화를 고려하는 경우, 필요한 실제 실행시간은 다음 (11)과 같이 계산된다.

$$\Theta' = \lceil AB \cdot \Pi \rceil \tag{11}$$

따라서, 양자화로 인한 오버헤드는 다음(12)와 같이

정의할 수 있다.

$$Q(\Pi,\Theta)=\left(\Theta'-\Theta\right)/\Pi \quad (12)$$

본 논문에서는 EDF의 경우에 대하여 양자화 오버헤드가 심각하게 발생할 수 있음을 보이고, 기존의 CPU 요구량이 실제 스케줄링 파라미터에 따라 어떻게 변하는지를 살펴보도록 한다.

3.3 양자화 오버헤드의 한계값

양자화 오버헤드는 주어진 주기에 대해 추상화 임계값이 정확하게 표현되지 못하는 경우에 대해 발생하는 것으로, 주기가 길어지면, 점차로 작아짐을 알 수 있다. 양자화 오버헤드의 한계값에 대한 함수를 다음과 같이 표현할 수 있다.

Lemma 1. 태스크 그룹의 양자화로 인한 오버헤드는 (13)과 같은 한계함수를 가진다.

$$Q(\Pi,\Theta)\le 1/\Pi \quad (13)$$

위 내용의 증명은 간단하므로, 지면관계상 생략한다. 이 때, 양자화 한계함수는 다음과 같이 나타낼 수 있다.

$$\Phi(\Pi)=1/\Pi \quad (14)$$

3.4 태스크 그룹의 실행 주기에 따른 추상화 임계값의 변화

추상화 오버헤드는 주기에 따라 변화하는 함수이다. 하지만, 이전 연구에서는 주기에 따른 직접적인 변화가 관계식으로 표현되지 않아 범위를 명확하게 알 수 없었다. 추상화 오버헤드와 태스크 그룹의 스케줄링 주기 간에는 다음과 같은 관계식이 성립한다.

Lemma 2. 추상화 오버헤드는 태스크 그룹의 스케줄링 주기와 다음 (15)와 같은 관계가 있다.

$$\frac{2U_w(1-U_w)}{(P_{\min}/\Pi)+2U_w+1}\le\Psi(\Pi)<\min\left(\frac{2U_w(1-U_w)}{(P_{\min}/\Pi)+2U_w-2},\frac{2U_w(1-U_w)}{2U_w+1}\right) \quad (15)$$

증명: 주기-배수 관계의 정의에 의해, 태스크 그룹 내 스케줄러가 EDF인 경우, 다음 (16)와 같은 관계식이 성립한다.

$$k+1<\frac{P_{\min}}{\Pi}+\frac{\Theta}{\Pi}\cdot\frac{2(k+1)}{k+2} \quad (16)$$

K는 위 식을 만족하는 최대 정수이므로, 다음과 같이 정리할 수 있다.

i) 만약 $\frac{\Theta}{\Pi}\cdot\frac{2(k+1)}{k+2}<1$ 이면,

$$k<\frac{P_{\min}}{\Pi}-\alpha \quad (0<\alpha\le 1)$$

$$k=\left\lfloor\frac{P_{\min}}{\Pi}\right\rfloor-1 \quad \text{or} \quad \left\lfloor\frac{P_{\min}}{\Pi}\right\rfloor. \quad (17)$$

ii) 만약 $\frac{\Theta}{\Pi}\cdot\frac{2(k+1)}{k+2}\ge 1$ 이면,

$$k<\frac{P_{\min}}{\Pi}+\alpha' \quad (0<\alpha'\le 1)$$

$$\left(\because\frac{\Theta}{\Pi}\cdot\frac{2(k+1)}{k+2}<2\right)$$

$$k=\left\lfloor\frac{P_{\min}}{\Pi}\right\rfloor \quad \text{or} \quad \left\lfloor\frac{P_{\min}}{\Pi}\right\rfloor+1 \quad (18)$$

위의 (17), (18)로부터, k의 범위를 얻을 수 있다.

$$k=\left\{\left\lfloor\frac{P_{\min}}{\Pi}\right\rfloor-1,\left\lfloor\frac{P_{\min}}{\Pi}\right\rfloor,\left\lfloor\frac{P_{\min}}{\Pi}\right\rfloor+1\right\}$$

$$\frac{P_{\min}}{\Pi}-2<k\le\frac{P_{\min}}{\Pi}+1 \quad (19)$$

추상화 오버헤드의 정의(9)로부터 추상화 임계값의 범위를 다음과 같이 구할 수 있다.

$$\frac{2U_w(1-U_w)}{(P_{\min}/\Pi)+2U_w+1}\le\Psi(\Pi)<\frac{2U_w(1-U_w)}{(P_{\min}/\Pi)+2U_w-2} \quad (20)$$

K의 최소값이 1임을 고려하여, k의 범위를 다음과 같이 정리할 수 있다.

$$\frac{2U_w(1-U_w)}{(P_{\min}/\Pi)+2U_w+1}\le\Psi(\Pi)<\min\left(\frac{2U_w(1-U_w)}{(P_{\min}/\Pi)+2U_w-2},\frac{2U_w(1-U_w)}{2U_w+1}\right) \quad (21)$$

위의 관계는 다음과 같이 다시 정리할 수 있다.

Corollary 1. 추상화 오버헤드는 태스크 그룹의 스케줄링 주기에 대해 다음과 같은 관계식으로 표현할 수 있다.

$$\Psi(\Pi) \le \begin{cases} \dfrac{2U_w(1-U_w)}{2U_w+1} & \text{if } \Pi \ge \dfrac{P_{\min}}{3}, \\ \dfrac{2U_w(1-U_w)}{(P_{\min}/\Pi)+2U_w-2} & \text{otherwise.} \end{cases} \quad (22)$$

증명: i) 만약 $P_{\min} \ge 3\Pi$ 라면, 일반적으로 다음 식을 유도할 수 있다.

$$\frac{P_{\min}}{\Pi} + 2U_w - 2 \ge 2U_w + 1$$

따라서, Lemma 2의 오른쪽 부등호 조건은 다음과 같이 다시 쓸 수 있다.

$$\min\left(\frac{2U_w(1-U_w)}{(P_{\min}/\Pi)+2U_w-2}, \frac{2U_w(1-U_w)}{2U_w+1}\right) = \frac{2U_w(1-U_w)}{2U_w+1}. \quad (23)$$

ii) 만약 $P_{\min} < 3\Pi$ 라면, 일반적으로 다음식이 성립한다.

$$\frac{P_{\min}}{\Pi} + 2U_w - 2 < 2U_w + 1$$

따라서, Lemma 2의 오른쪽 부등호 조건은 다음과 같이 다시 쓸 수 있다.

$$\min\left(\frac{2U_w(1-U_w)}{(P_{\min}/\Pi)+2U_w-2}, \frac{2U_w(1-U_w)}{2U_w+1}\right) = \frac{2U_w(1-U_w)}{(P_{\min}/\Pi)+2U_w-2}. \quad (24)$$

Lemma 1과 Corollary 2를 이용하여, 다음과 같은 정리를 이끌어낼 수 있다.

Theorem 1. 양의 정수 α^* 에 대하여, 추상화 오버헤드와 양자화 오버헤드 간에 다음과 같은 관계식을 만족하는 α^*가 존재한다.

$$\forall \alpha \ge \alpha^* \quad \Phi(\Pi) - \Phi(\Pi+\alpha) > 0, \quad \text{and} \quad \Psi(\Pi) - \Psi(\Pi+\alpha) \le 0. \quad (25)$$

i) 모든 양의 정수 α 에 대하여,

$$\Phi(\Pi) > \Phi(\Pi+\alpha). \quad (26)$$

$$\left(\because \quad \forall \alpha > 0, \quad \Phi(\Pi) - \Phi(\Pi+\alpha) = \frac{1}{\Pi} - \frac{1}{\Pi+\alpha} > 0\right)$$

ii) Lemma 2 로부터 다음 (27)을 얻을 수 있다.

$$\Psi(\Pi+\alpha) \ge \frac{2U_w(1-U_w)}{P_{\min}/(\Pi+\alpha)+2U_w+1}. \quad (27)$$

또한, Corollary 1과 위 식으로부터,

a) 만약 $\Pi < P_{\min}/3$ 인 경우, 다음 (28)과 같은 관계식을 얻을 수 있다.

$$\Psi(\Pi) - \Psi(\Pi+\alpha) < \frac{2U_w(1-U_w)}{P_{\min}/\Pi+2U_w-2} - \frac{2U_w(1-U_w)}{P_{\min}/(\Pi+\alpha)+2U_w+1}. \quad (28)$$

위 식(28)은 양의 상수 c에 대해, 다음과 같이 정리할 수 있다.

$$2U_w(1-U_w)\cdot\left(\frac{1}{\frac{P_{\min}}{\Pi}+2U_w-2} - \frac{1}{\frac{P_{\min}}{\Pi+\alpha}+2U_w+1}\right) = c\cdot(3\Pi - P_{\min})\alpha + 3\Pi^2 \quad (29)$$

따라서, $\alpha > \dfrac{3\Pi^2}{P_{\min}-3\Pi}$ 인 α 에 대하여

$$\Psi(\Pi) - \Psi(\Pi+\alpha) < 0 \quad (30)$$

이므로, (25)를 만족하는 α^*가 존재한다.

b) 만약 $\Pi \ge P_{\min}/3$인 경우 다음 식을 얻는다.

$$\Psi(\Pi) - \Psi(\Pi+\alpha) < \frac{2U_w(1-U_w)}{2U_w+1} - \frac{2U_w(1-U_w)}{\frac{P_{\min}}{\Pi+\alpha}+2U_w+1} \quad (31)$$

위의 (31)에서 $\Psi(\Pi)$ 는 $\dfrac{2U_w(1-U_w)}{2U_w+1}$ 로 점근한다.

따라서, 매우 큰 α^*에 대해 다음 (32)식이 성립한다.

$$\Psi(\Pi) \le \Psi(\Pi+\alpha) \quad (32)$$

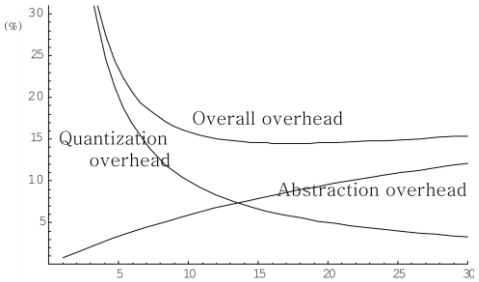

그림 1 양자화 오버헤드, 추상화 오버헤드가 전체 오버헤드에 미치는 영향

3.5 정리 1의 해석

Theorem 1은 양자화 오버헤드의 한계함수가 감소하는 구간에 대하여, 추상화 오버헤드가 증가하는 구간이 반드시 존재함을 표현하고 있다. 즉, 추상화 오버헤드는 주기의 증가에 따라 증가하는데 반해, 양자화 오버헤드의 한계값은 주기가 증가함에 따라 감소한다.

이러한 현상에 대한 직관적인 해석은 다음과 같다. 양자화 오버헤드의 경우, 주기가 증가하면 보다 세밀하게 양자화하여 CPU 요구량을 표시할 수 있기 때문에 오버헤드의 한계값은 감소하게 된다. 반면, 주기가 증가함에 따라 주기적 실행에 필요한 최악 수행시간은 증가하므로, 주어진 t시간에 대하여, 자원 제공함수는 감소하게 된다. 이는 곧 추상화 임계값과 추상화 오버헤드에 영향을 미치게 된다.

이는 CPU 처리량의 양자화로 추상화 한계값과 다른 CPU 요구량을 필요로 함을 의미한다. 두 가지 오버헤드를 동시에 고려한 태스크 그룹에 대한 CPU 요구량 함수는 기존의 추상화 한계 함수와 다르게 나타난다. 예제를 통해 구체적인 사례에 대해 이해해 볼 수 있다. 태스크 그룹 G가 T1(50,7)과 T2(75,9)의 두 주기 태스크를 실행하고 있다고 가정하자. 또한, 로컬 스케줄러는 EDF를 사용하고 있다고 가정한다.

이 때, 추상화 오버헤드, 양자화 오버헤드와 전체 오버헤드는 그림 1과 같이 나타난다. 그림에서 주기 변화에 따라 두 오버헤드는, Theorem 1에서 본 바와 같이, 반비례적인 성향을 가지고 있음을 알 수 있다.

4. 결론 및 추후 연구

본 논문에서는 대표적인 계층형 실시간 스케줄러인 조합형 스케줄러의 문제에 대하여 살펴보았다. 조합형 스케줄러는 여러 태스크들을 하나의 그룹으로 묶어 계층적으로 스케줄링 할 수 있는 방법을 제공한다. 하지만, 주기적인 자원할당에 있어 실제 시스템에서 CPU 처리량을 분배하여 제공하는 단위인 틱-단위 양자화를 고려하지 않아, 정확한 CPU 요구량을 얻을 수 없었다. 따라서, 본 논문에서는 양자화에 따른 조합형 스케줄러의 변화와 영향을 분석하였다. 분석의 결과 양자화 오버헤드와 기존의 추상화 오버헤드가 반비례하는 관계에 있음이 밝혀졌다.

결과적으로, 이러한 양자화에 대한 문제는 계층형 실시간 스케줄러의 CPU 요구량 함수를 변화시킨다. 이전의 추상화 한계 함수가 주기와 관련하여, 비교적 단순한 증가형 그래프였으나, 단조 증가/감소 그래프가 아님이 밝혀짐에 따라 최적의 CPU 요구량을 다시 정확하게 찾아야 할 필요성이 제기된다. 즉, 최소의 CPU 요구량을 가지는 스케줄링 파라미터 (주기, 실행시간)을 찾을 수 있는 알고리즘이 추가로 연구되어야 한다. 또한, 현재 증명은 로컬 스케줄러가 EDF인 경우에만 유효하므로, RM의 경우에 대하여 추가적인 분석이 필요하다.

참고문헌

1. Shin, I. and Lee, I. 2008. Compositional real-time scheduling framework with periodic model. ACM Trans. Embed. Comput. Syst. Vol. 7, No. 3 pp.1-39. (Apr. 2008), DOI= http://doi.acm.org/10.1145/1347375.1347383

2. Shin, I. and Lee, I. 2003. Periodic Resource Model for Compositional Real-Time Guarantees. In Proceedings of the 24th IEEE international Real-Time Systems Symposium (December 03 - 05, 2003). RTSS. IEEE Computer Society, Washington, DC, 2.

3. Easwaran, A., Anand, M., and Lee, I. 2007. Compositional Analysis Framework Using EDP Resource Models. In Proceedings of the 28th IEEE international Real-Time Systems Symposium (December 03 - 06, 2007). RTSS. IEEE Computer Society, Washington, DC, 129-138. DOI=http://dx.doi.org/10.1109/RTSS.2007.17

4. Easwaran, A., Shin, I., and Lee, I. 2009. Optimal virtual cluster-based multiprocessor scheduling. Real-Time Syst. 43, 1 (Sep. 2009), 25-59. DOI= http://dx.doi.org/10.1007/s11241-009-9073-x

5. Feng, X. and Mok, A. K. 2002. A Model of Hierarchical Real-Time Virtual Resources. In Proceedings of the 23rd IEEE Real-Time Systems Symposium (December 03 - 05, 2002). RTSS. IEEE Computer Society, Washington, DC, 26.

6. Mok, A. K., Feng, X. , and Chen, D. 2001. Resource Partition for Real-Time Systems. In Proceedings of the Seventh Real-Time Technology and Applications Symposium (RTAS '01) (May 30 - June 01, 2001). RTAS. IEEE Computer Society, Washington, DC, 75.

7. Baruah, S. K., Cohen, N. K., Plaxton, C. G., and Varvel, D. A. 1993. Proportionate progress: a notion of fairness in resource allocation. In Proceedings of the Twenty-Fifth Annual ACM Symposium on theory of Computing (San Diego, California, United States, May 16 - 18, 1993). STOC '93. ACM, New York, NY, 345-354. DOI= http://doi.acm.org/ 10.1145/167088.167194

8. Anderson, J. H. and Srinivasan, A. 2000. Pfair scheduling: beyond periodic task systems. In Proceedings of the Seventh international Conference on Real-Time Systems and Applications (December 12 - 14, 2000). RTCSA. IEEE Computer Society, Washington, DC, 297.

9. Baruah, S. K., Gehrke, J., and Plaxton, C. G. 1995. Fast scheduling of periodic tasks on multiple resources. In Proceedings of the 9th international Symposium on Parallel Processing (April 25 - 28, 1995). IPPS. IEEE Computer Society, Washington, DC, 280-288.

10. Anderson, J. H. and Srinivasan, A. 2004. Mixed Pfair/ERfair scheduling of asynchronous periodic tasks. J. Comput. Syst. Sci. 68, 1 (Feb. 2004), 157-204. DOI= http://dx.doi.org/10.1016/j.jcss.2003.08.0 02