실시간 시스템의 주기적 태스크의 최적 오프셋 탐색

An Approach to Optimize Initial Offsets of Periodic Tasks in Real-Time Systems

요 약

실시간 시스템(real-time system)은 논리적 연산을 일정한 시간적 제약 하에서 수행하는 시스템이다. 시간적 제약을 충족하도록 주기적 태스크(periodic task)를 스케줄(schedule)할 때 일반적으로 태스크 오프셋(initial offset)이 0 이거나 고정된 것으로 가정한다. 그러나 오프셋에 약간의 유연성을 허용함으로써 태스크들의 평균 응답 시간을 줄일 수도 있다. 이 논문에서는 주기적 태스크의 오프셋을 주어진 허용 범위 안에서 선택하여 평균 응답 시간(response time)을 최적화할 수 있음을 보이고, 임의의 태스크 집합에 대하여 최적 오프셋이 존재하는 좁은 범위를 제시한다.

1. 서 론

실시간 시스템(real-time system)은 논리적 정확성과 시간적 제약의 충족이 동시에 요구되는 시스템이다. 목적에 따라 GPS 항해 시스템, 실시간 멀티미디어 서비스 시스템, 레이더(radar) 시스템, 스마트 홈 시스템 등 다양한 형태의 실시간 시스템이 도입되었으며, 활용 분야가 계속 확대되고 있다. 이러한 실시간 시스템들은 수행하는 구체적인 과제는 서로 다르지만, 정해진 시간적 제약을 준수해야 한다는 공통점이 있다[1].

실시간 시스템의 시간적 제약은 구체적으로 각 과제(job)에 대한 데드라인(deadline)으로 주어진다. 일정한 주기(period)에 따라 릴리스(release)되는 과제들의 집합을 주기적 태스크(periodic task)라고 한다. GPS 항해 시스템은 주기적으로 GPS 위성의 신호를 수신하여 사용자의 현재 위치를 계산한다. 이와 같이 작은 과제를 주기적으로 반복하는 많은 실시간 시스템들은 주기적 태스크 모델로 설명할 수 있다.

태스크의 집합이 주어지면 시스템은 그 집합의 모든 태스크가 시간적 제약을 준수하도록 프로세서와 자원을 할당하여야 한다. 주기적 태스크 모델에서 프로세서의 선점(preemption)을 허용하는 고정 우선순위 기반의 스케줄(fixed-priority-driven schedule)의 경우, 모든 태스크의 첫 번째 과제의 릴리스 타임인 오프셋(initial offset)이 0 일 때 선점 시간이 최대가 된다. 하나의 과제가 릴리스 된 시점부터 수행을 마치는 시점까지의 시간을 응답 시간(response time)이라 하는데, 모든 태스크의 오프셋이 0 이면 각 태스크의 첫 번째 과제는 최악의 응답 시간을 가지는 것이다. 여기에서 최악의 응답 시간이란, 주어진 태스크의 주기와 수행시간(execution time)을 고려하였을 때 각 태스크의 과제가 가질 수 있는 가장 긴 응답 시간이라는 뜻이다. 이러한 관찰을 바탕으로, 전통적인 스케줄 가능성(schedulability) 검사는 모든 태스크의 오프셋이 0 인 경우에 각 태스크의 첫 번째 과제가 스케줄 가능하면, 이후의 과제들도 가능하며 태스크들의 오프셋이 변하여도 스케줄 가능하다고 한다[2]. 그러나, 모든 태스크의 오프셋이 0 이라면, 주기들의 공배수인 하이퍼 주기 (hyper-period)마다 릴리스되는 과제들은 필연적으로 최악의 응답 시간을 가지게 된다[3].

이 연구는 실시간 시스템의 스케줄 가능한 주기적 태스크의 최적 오프셋을 탐색한다. 각 태스크의 오프셋을 일정한 범위 안에서 선택할 수 있다면, 최적 오프셋을 선택함으로써 태스크들의 평균 응답 시간을 최소화 할 수 있다. 예를 들어, 실시간 멀티미디어 서비스 시스템은 사용자로부터 서비스 요청을 받으면 주기적으로 멀티미디어 파일을 압축해서 사용자에게 전송한다. 이 때 사용자는 첫 번째 파일을 최대한 빨리 받는 것 보다 평균적으로 가장 최신 정보를 받는 것을 선호할 것이다. 이와 같이 태스크의 오프셋에 약간의 유연성(flexibility)이 허용될 때, 평균 응답 시간을 최소화하여 시스템의 반응성(responsiveness)을 극대화하고 견고성(robustness)의 향상에 기여하는 것이 이 연구의 목표이다.

관련된 연구로는 태스크의 오프셋이 0 이 아닌 경우의 스케줄 가능성 검사에 대한 연구와[4], 오프셋을 상대적으로 자유롭게 선택할 수 있을 때 스케줄 가능성이 가장 높아지도록 오프셋들을 결정하는

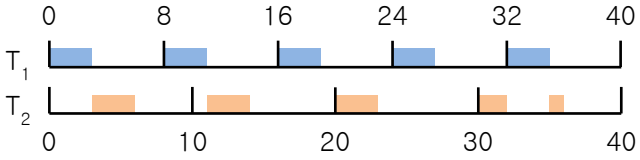

그림 1 두 태스크의 오프셋 차이가 0 인 경우

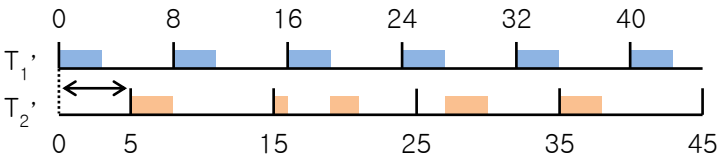

그림 2 두 태스크의 오프셋 차이가 5 인 경우

방법을 제시한 연구가 있다[5]. 첫 번째 연구[4]에서는 고정된 오프셋에 대해 검사를 하였지만 이 연구에서는 오프셋을 약간 변화시켜서 상황을 향상시키는 것에 시도한다. 그리고 두 번째 연구[5] 에서는 모든 태스크의 주기가 동일한 경우에만 스케줄 가능성이 높은 오프셋을 구하였는데, 이 연구에서는 태스크 주기에 제한이 없는 일반적인 경우에 대하여 오프셋 최적화를 추구한다.

2. 시스템 모델

이 연구에서는 다음과 같은 주기적 태스크 시스템 모델을 가정한다.

1) 이 시스템에 주어진 일은 n 개의 주기적 태스크 $T_1, \cdots, T_n$ 이다. $(1 \le n)$
2) 각 T_i 는 오프셋 O_i, 주기 P_i 를 가지며, T_i 의 모든 과제는 수행 시간 E_i, 상대적 데드라인 D_i 를 가진다. $(1 \le i \le n)$
3) T_i 의 j 번째 과제 J_{ij} 의 릴리스 타임은 R_{ij} 로 나타내며, 그 값은 다음과 같다. $(1 \le j)$
$$R_{ij} = O_i + P_i \times (j - 1)$$
4) J_{ij} 의 수행을 마치는 시점을 F_{ij} 라고 하면, J_{ij} 의 응답 시간 S_{ij} 는 다음과 같다.
$$S_{ij} = F_{ij} - R_{ij}$$
5) 이 논문에서는 설명의 단순화를 위하여 $P_i = D_i$ 이고, 프로세서는 하나이며 선점을 허용하고, 태스크들은 서로 독립이고, 고정 우선순위 기반의 스케줄을 하는 경우를 가정한다. 이 경우에는 상대적으로 짧은 주기의 태스크에 높은 우선순위를 부여하는 비율 단조(rate monotonic) 스케줄이 최적 스케줄이다[1]. 이후의 논의는 완화된 가정 하에서도 성립한다.

3. 평균 응답 시간 분석

이 장에서는 태스크 집합의 오프셋을 변화시켜서 평균 응답 시간이 줄어드는 경우를 확인한다. n = 2 인 간단한 예를 살펴보자. T_1의 (P_1, E_1)은 (8, 3), T_2의 (P_2, E_2)는 (10, 3)이고,
$$O_1 = O_2 = 0$$
이라고 하자. 그림 1은 $\{T_1, T_2\}$의 비율 단조 스케줄을 가로축인 타임라인(timeline) 위에 나타낸 것이다. 이 태스크 집합은 오프셋이 모두 0 일 때 스케줄 가능하므로 오프셋이 어떤 값을 가져도 스케줄 가능하다. 이 집합의 평균 응답 시간을 분석하기 전에 다음 정의와 보조정리 1을 살펴보자.

정의

태스크 T_a 의 과제 J_{aj} 와 T_b 의 과제 J_{bk} 의 릴리스 타임 오프셋 O(a, j, b, k)는 다음과 같이 정의한다:
$$O(a, j, b, k) = R_{bk} - R_{aj}.$$

보조정리 1

임의의 태스크 집합 $\{T_1, \cdots, T_m\}$ $(1 \le m)$의 릴리스 타임 오프셋은 주기 $P_1, \cdots, P_m$ 의 최소공배수인 하이퍼 주기를 기준으로 반복된다. 따라서, 태스크의 스케줄도 하이퍼 주기를 기준으로 반복된다[3].

$\{T_1, T_2\}$의 하이퍼 주기는
$$LCM(P_1, P_2) = LCM(8, 10) = 40$$
이므로, 보조정리 1에 의해 전체 구간에서의 T_2 의 응답 시간 $\{S_{2j} \mid 1 \le j\}$은 구간 [0, 40) 에서의 응답 시간 $\{S_{21}, S_{22}, S_{23}, S_{24}\}$의 반복이다. 구간 [0, 40) 에서의 T_2의 평균 응답 시간 S_2 는 다음과 같다.
$$S_2 = (S_{21} + S_{22} + S_{23} + S_{24}) / 4$$
$$= (6 + 4 + 3 + 6) / 4 = 4.75$$
이제 T_1, T_2 의 오프셋을 다음과 같이 바꾼 태스크를 T_1', T_2' 이라고 하자.
$$O_2 - O_1 = 5$$
그림 2는 $\{T_1', T_2'\}$의 비율 단조 스케줄을 나타내며, 구간 [0, 40) 에서의 T_2' 의 평균 응답 시간 S_2' 은 다음과 같이 표현할 수 있다.
$$S_2' = (S_{21}' + S_{22}' + S_{23}' + S_{24}') / 4$$
$$= (3 + 6 + 5 + 3) / 4 = 4.25$$
두 태스크의 오프셋 차이$(O_2 - O_1)$를 0 에서 5 로 바꾼 결과, 태스크 T_1 의 평균 응답 시간에는 변화가 없고 우선순위가 낮은 태스크 T_2 의 평균 응답 시간은 0.5 감소하였다.

4. 오프셋의 범위 분석

앞 장에서 태스크 집합의 오프셋을 바꾸어서 평균 응답 시간을 감소시킬 수 있음을 보았다. 각 태스크의 주기와 오프셋의 범위가 주어지면, 보조정리 1에 의해, 주어진 오프셋의 범위와 하이퍼 주기 안에서 평균 응답

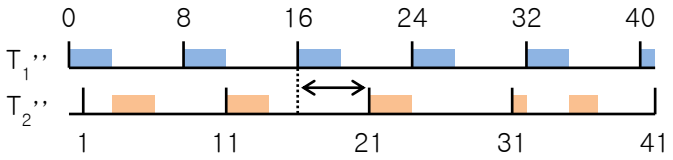

그림 3 두 태스크의 오프셋 차이가 1 인 경우

시간을 최소화하는 오프셋이 최적의 오프셋일 것이다. 그러나 하이퍼 주기의 크기가 커지면 오프셋 탐색 공간이 커져서 탐색 비용이 증가할 수 있다. 이 장에서는 최적 오프셋의 평균 응답 시간을 유지하면서 오프셋 탐색 공간을 축소한다.

아래에 제시된 정의는 두 태스크 집합이 초기의 일정한 시간 이후부터 같은 수행 스케줄을 가지면 그 두 태스크 집합은 동등하다고 표현하고 있다. 정리 1은 임의의 2-태스크 집합에 대해 오프셋의 차이가 두 태스크의 주기의 GCD 보다 작으면서 원래의 집합과 동등한 집합이 존재한다고 말하고 있다. 즉, 최적의 오프셋은 태스크 주기들의 GCD 범위 안에서 찾을 수 있다는 것이다. 정리 1을 앞 장의 $\{T_1', T_2'\}$에 적용하면 두 태스크의 오프셋 차이를 5 로부터

$$5 \bmod GCD(8, 10) = 1$$

로 줄여서 $\{T_1', T_2'\}$와 동등한 태스크 집합 $\{T_1'', T_2''\}$을 얻게 된다. 그림 3은 $\{T_1'', T_2''\}$의 비율 단조 스케줄을 나타내고 있다. 그림 2에서 양쪽 화살표(↔)로 표시된 부분인 오프셋이 그림 3에서 O(1, 3, 2, 3)과 같으므로, 두 태스크 집합이 동등하다는 것을 알 수 있다. 따름정리 1은 정리 1의 내용을 여러 개의 태스크를 가진 집합에 대하여 확장하고 있다.

정의

하나의 태스크 집합 X에 대해 시점 t에 유효한 과제(릴리스 타임이 t 이전이고, 데드라인이 t 이후인 과제) 전체의 집합을 $Eff_t(X)$ 라고 하자. 두 n-태스크 집합 A와 B에 대하여, A의 임의의 태스크의 성질 중 오프셋을 제외한 모든 나머지(주기, 수행시간 등)를 그대로 가지는 태스크가 B에 포함되어 있고, 그 반대도 성립하며, 다음을 만족하는 시점 t1, t2가 존재한다면, A와 B는 동등하다(equivalent)고 말한다:

$$\forall J_{aj}, J_{bk} \in Eff_{t1}(A), \exists J_{av}, J_{bw} \in Eff_{t2}(B)$$
$$\text{such that } O(a,j : b,k) = O(a,v : b,w).$$

정리 1

각각 주기가 P_x, P_y인 태스크 T_x, T_y로 이루어진 태스크 집합에서 두 태스크의 오프셋 차이가 α 이면, 이 집합과 동등하고 오프셋 차이가

$$\beta = \alpha \bmod GCD(P_x, P_y)$$

인 태스크 집합이 존재한다.

(증명) 유클리드 알고리즘을 활용한다.

따름정리 1

주기가 P_1, … , P_z인 태스크 T_1, … , T_z의 집합과 동등하며 오프셋 차이가 다음보다 작은 집합이 있다:

$$LCM\{GCD(P_1, P_y) \mid 1 < y \le z\}.$$

5. 결론 및 향후 연구

이 논문에서는 실시간 시스템의 주기적 태스크 집합의 평균 응답 시간을 최적화하는 태스크 오프셋에 대하여 분석하고 탐색하였다. 3.에서는 태스크들의 하이퍼 주기 안에 최적 오프셋이 존재함을 보였고, 4.에서는 최적 오프셋 탐색 공간을 태스크 주기들의 GCD 범위로 축소하였다.

축소된 탐색 공간 안에서 최적 오프셋을 결정하기 위해서 현재는 대규모 탐색, 선형 프로그래밍, 확률적 방법 등을 활용할 수 있다. 태스크들 사이의 관계를 분석하여 보다 효율적이고 정확하게 최적 오프셋을 결정하는 방법을 구하는 것이 앞으로 연구할 과제이다.

참 고 문 헌

[1] Jane W. S. Liu, “Real-Time Systems,” Prentice Hall, 2000.

[2] Lui Sha et al., “Real Time Scheduling Theory: A Historical Perspective,” in *Real-Time Systems*, vol. 28, 2004.

[3] Ismael Ripoll, Alfons Crespo and Aloysius K. Mok, “Improvement in feasibility testing for real-time tasks,” in *Real-Time Systems*, vol. 11, 1996.

[4] Rodolfo Pellizzoni and Giuseppe Lipari, “A New Sufficient Feasibility Test for Asynchronous Real-Time Periodic Task Sets,” in *Proceedings of the 12th 16th Euromicro Conference on Real-Time Systems*, 2004.

[5] I. Bate and A. Burns, “Schedulability Analysis of Fixed Priority Real-Time Systems with Offsets,” in *Proceedings of the 9th Euromicro Workshop on Real-Time Systems*, 1997.