입/출력 장치의 소비전력 최적화를 위한 타스크 스케쥴링

정도한
과학영재학교
김태환[0]
서울대학교, 전기.컴퓨터공학부
{dchung, tkim}@ssl.snu.ac.kr

Optimal Task Scheduling for Minimizing Energy Consumption in I/O Devices

Dohan Jeong
Busan Science Academy
Taewhan Kim[0]
School of EECS, Seoul National University

요 약

임베디드 시스템에서 입출력에서 소모되는 전력양은 상당한 수준이다. 입출력 장치에서의 수행되는 타스크의 순서를 정하여 전체적으로 입출력 장치에서의 휴식 시간을 최대한 많이 허락하는 기법이 필요하다. 기존의 연구는 입출력 장치 소비 전력 최소화를 위한 타스크 스케쥴링 문제를 단순한 지협적인 휴리스틱에 기반하여 풀었다. 본 연구는 기존의 연구에서의 방법과는 달리 최적의 해를 내는 해법을 제시한다, 구체적으로 시간 제약 조건을 가진 저전력을 위한 타스크 스케쥴링 문제를 ILP (integer linear programming) 기법을 적용하는 방법을 제시한다. 본 연구는 또한, 실험을 통해 주어진 시간 안에 최적의 해를 구하는 문제의 크기를 판단하는 기준을 제시할 수 있다는 의의를 가진다.

1. 서 론

여러 가지 전자기기의 시장이 넓어지고 있는 현실에서, 그 중 아주 놀라운 것이 모바일 기기의 발달이다. 모바일 기기란 휴대하기 편리한 기기들을 말하는 것으로서 휴대용 전화기, PDA, 등이 있다. 이런 기기들은 과거에는 그렇지 않던 것이 점점 발달하여 요즈음에 들어서는 그 작고 휴대하기 편리한 기기로 엄청나게 많은 일들을 할 수 있게 되었다. 예를 들어, 휴대용 전화기로 인터넷이나 게임을 즐기는 등의 일이다. 이렇게 모바일 기기가 편리해 지고 있다. 하지만 이 모바일 기기에서 중요하게 고려되어야 할 점들이 있다. 바로 메모리 용량 문제와 속도, 배터리의 문제 등이다. 모바일 기기의 특성상 배터리를 큰 것을 달아서 오래 가게 할 수도 없는 노릇이며 커다란 메모리를 달아 용량을 늘릴 수도 없는 노릇이다. 이 논문에서는 그 고려해야 할 사항들 중 배터리의 문제에 대해 따져보기로 하자.

이 점은 모바일 기기에서는 아주 중요한 문제이다. 이 점에 대해서는 앞에서 말했듯이 배터리를 크게 만들어 단다던가 하는 등의 작업은 모바일 기기의 특성상 불가능하다고 이미 밝혔었다. 그렇다면 이 배터리의 문제는 어떻게 해결할 것인가? 이 물음에 대해 여러 가지 해답이 나올 수 있다. 성능을 약간 떨어지게 한다든지의 처리이다. 하지만 사용자는 언제나 최상의 품질을 원한다. 이런 점을 감안 했을 때 이런 방법도 가능하지 않다고 생각한다. 그러므로 다른 방법을 강구해 봐야 한다.

모바일용 임베디드 시스템에서 특히 전력 소모를 많이 가지는 부분이 입출력 장치에 있다고 보고되고 있다. 입출력에서 수행되는 타스크 (또는 작업)의 *스케쥴링* (scheduling)은 입출력 장치의 휴식 시간 조절과 일접한 관계가 있으며, 이는 입출력 장치에서 소모되는 전력량을 결정한다. 예를 들면, 어떤 데이터를 읽고 쓰기위해 메모리에 접근하는 과정에서 스케쥴링을 이용한 것이다. 즉, 메모리에 접근하는 순서를 적절히 하여 *소비전력*을 감소시킨 것이다. 다음으로는 작업들을 스케쥴링하여 그 작업에서 필요한 Device들을 *소비전력*이 적게드는 off상태로 가능한 만큼 최대한 많이 만들어 *소비전력*을 감소시키는 방법이다.

이 논문은 저전력 입출력 작업 스케쥴링에 관한 문제에 대한 연구이다. 기존의 연구는 입출력 장치 소비 전력 최소화를 위한 타스크 스케쥴링 문제를 단순한 지협적인(local) 휴리스틱에 기반하여 풀었다 [1,2]. 본 연구는 기존의 연구에서의 방법과는 표현 및 문제 접근 방법이 다른, 즉, 먼저 최적의 해를 내는 해법을 ILP (Integer linear programming) 공식화를 통해 실현하며, 본 연구의 또 하나의 가치는, 주어진 시간 안에 최적의 해를 구하는 문제의 크기를 판단하는 기준을 제시하는데 이용될 수도 있다.

2. 최적해를 위한 ILP 공식화 기법

2.1 문제 정의

총 n개의 job (타스크 또는 작업)들이 주어져 있다. 각각의 job들에게는 job을 수행하는데 필요한 device들이 한 개씩 있다. 그 device가 결정적으로 job을 수행하며 전력을 소비한다. 그리고 각 job들은 꼭 끝마쳐야하는 시간이 있으며 수행되는 데에도 각각 다른 시간이 걸린다. 즉, deadline이 되는 시간이 있으며 수행시간이 있다는 것이다. job i의 deadline은 d_i로 표시하고, job i의 수행시간은 c_i로 표시한다. 그리고 특별히 job들의 deadline의 최대 값, 즉 어떤 job이든 그 deadline이 지나고서는 실행이 되어서는 안 되는 deadline을 T라고 하고 이것을 총 시간이라고 한다.

각 job들의 Device들은 4가지 상태일 수 있다. (1) 켜져 있는 상태, (2) 꺼져 있는 상태, (3) 켜지고 있는 상태, (4) 꺼지고 있는 상태가 그것이다. 그리고 job을 수행할 수 있는 상태는 켜져 있는 상태이다. 또, 시작할 때 모든 device들은 켜져 있다고 가정한다. 각 상태에서는 소비되는 전력의 양도 각기 다르다. 그러므로 각 상태들을 총 시간 T동안 잘 조합해 *소비전력*을 *최소화* 시켜야 한다. 물론 각 job의 deadline은 잘 지켜져야 하며, 상태 전이를 할 때 가능하게 해야 한다. 즉, 켜져 있는 상태에서 바로 꺼져 있는 상태로 갈 수 없고, 중간에 꺼지고 있는 상태를 거쳐야 한다는 것이다.

각 job의 device가 켜져 있는 상태에서 단위시간동안 소비하는 전력의 양을 P_{wi}라고 하고, 꺼져 있는 상태에서 단위시간동안 소비하는 전력의 양을 P_{si}라고 하고, 켜지고 있는 상태에서 단위시간동안 소비하는 전력의 양을 P_{tui}라고 하며, 꺼지고 있는 상태에서 단위시간동안 소비하는 전력의 양을 P_{tdi}라고 한다. 여기서 단위시간이라 함은 총 시간 T를 T개의 시간으로 나누었을 때 하나의 시간을 말한다.

2.2 ILP 공식화

사용 변수들의 정의 : 단위시간 j에서 job i에 대해 5개의 변수를 지정하였다. 4개의 변수는 그 단위시간 j에서 job i의 device가 4가지 상태중의 하나인지 아닌지 로서 그 기준을 잡았으며 나머지 1개의 변수는 device가 켜져 있는 가운데서 job이 수행되고 있는지에 대한 것이다. 따라서 전에 언급한 4가지 상태를 나타내던 변수 중 켜져 있는지 아닌지를 나타내는 변수에는 무조건 job이 수행되고 있지는 않다는 전제하의 변수들이며 다음과 같다.

$$W_{ij} = \begin{cases} 1 & \text{job i가 time j에서 수행되지 않고 있고 job i에서 사용되는 device가 켜져 있는 상태 일 때} \\ 0 & \text{아닐 때} \end{cases}, \forall i, \forall j$$

$$S_{ij} = \begin{cases} 1 & \text{job i에서 사용되는 device가 단위시간 j에서 꺼져 있는 상태 일 때} \\ 0 & \text{아닐 때} \end{cases}, \forall i, \forall j$$

$$tu_{ij} = \begin{cases} 1 & \text{job i에서 사용되는 device가 단위시간 j에서 켜지고 있는 상태 일 때} \\ 0 & \text{아닐 때} \end{cases}, \forall i, \forall j$$

$$td_{ij} = \begin{cases} 1 & \text{job i에서 사용되는 device가 단위시간 j에서 꺼지고 있는 상태 일 때} \\ 0 & \text{아닐 때} \end{cases}, \forall i, \forall j$$

$$a_{ij} = \begin{cases} 1 & \text{job i가 단위시간 j에서 수행될 때} \\ 0 & \text{아닐 때} \end{cases}, \forall i, \forall j$$

목적함수의 정의 : 목적함수는 다음과 같다.

$$E = \min \sum_i E_i$$

E_i란 job i가 총 시간 T동안 소비한 전력의 양이며, 위 식은 모든 job들의 *소비전력*을 합한 것을 *최소화* 시키는 것이다. E_i는 다음과 같은 식으로 구해진다.

$$E_i = \sum_j (P_{wi} W_{ij} + P_{si} S_{ij} + P_{tui} tu_{ij} + P_{tdi} td_{ij} + P_{wi} a_{ij})$$

제약조건 :

(1) 하나의 단위시간에서 하나의 job의 device는 하나의 상태만을 지녀야 한다.

$$W_{ij} + S_{ij} + tu_{ij} + td_{ij} + a_{ij} = 1 \text{ for } \forall i, \forall j$$

(2) 총 시간동안 job이 수행된 단위시간의 수는 그 job의 수행시간과 같아야 한다.

$$\sum_j a_{ij} = C_i \quad \text{for } \forall i$$

(3) 그 job의 deadline 이후에는 job이 수행되어서는 안 된다.

$$\sum_{j=d_i+1}^{T} a_{ij} = 0 \quad , \forall i, \forall j$$

(4) 한 단위시간에서는 2가지 이상의 job이 수행되어서는 안 된다.

$$\sum_i a_{ij} \leqq 1 \quad \text{for } \forall j$$

(5) 어떤 단위시간에 job이 수행되기 위해서는 그 전의 단위시간에 이미 job이 수행되고 있거나 그냥 켜져 있는 상태이거나 켜지고 있는 상태이어야 한다.

$$a_{i(j-1)} + W_{i(j-1)} + tu_{i(j-1)} - a_{ij} \geqq 0 \quad \text{for } \forall i, \forall j$$

(6) 어떤 단위시간에 어떤 job의 device가 켜지고 있는 상태가 되려면 그 전의 단위시간에 꺼져 있는 상태이거나 꺼지고 있는 상태이어야 한다.

$$S_{i(j-1)} + td_{i(j-1)} - tu_{ij} \geqq 0 \quad \text{for } \forall i, \forall j$$

(7) 어떤 단위시간에 어떤 job의 device가 꺼져 있는 상태가 되기 위해서는 그 전의 단위시간에 켜지고 있는 상태이거나 이미 꺼져 있는 상태이어야 한다.

$$td_{i(j-1)} + S_{i(j-1)} - S_{ij} \geqq 0 \quad \text{for } \forall i, \forall j$$

(8) 어떤 단위시간에 어떤 job의 device가 켜지고 있는 상태가 되기 위해서는 그 전의 단위시간에 job을 수행하고 있는 상태이거나 그냥 켜져 있는 상태이거나 켜지고 있는 상태이어야 한다.

$$a_{i(j-1)} + tu_{i(j-1)} + W_{i(j-1)} - td_{ij} \geqq 0 \quad \text{for } \forall i, \forall j$$

(9) 어떤 단위시간에 어떤 job의 device가 그냥 켜져 있는 상태가 되기 위해서는 그 전의 단위시간에 job을 수행하고 있는 상태이거나 켜지고 있는 상태이거나 이미 그냥 켜져 있는 상태이어야 한다.

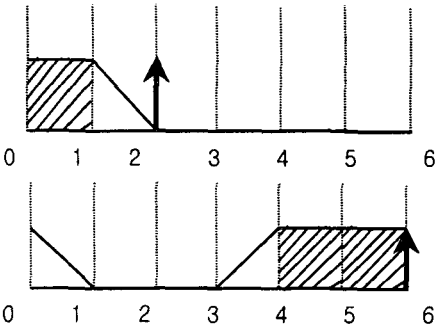
$$a_{i(j-1)} + tu_{i(j-1)} + W_{i(j-1)} - W_{ij} \geqq 0 \quad \text{for } \forall i, \forall j$$

3. 실험 결과

우리는 직접 ILP로 여러 변수들과 제약조건들을 이용하여 공식화 한 것을 이용해 직접 스케쥴링을 해야 하는 작업들의 데이터를 넣어보고 그 결과를 내어 보았다. [표-1]에 나타난 데이터에 대한 결과는 [그림-1]에 보여준다.

	c_i	d_i	P_{wi}	P_{si}	P_{tui}	P_{tdi}
job_1	1	2	4	1	3	2
job_2	2	6	5	2	4	3

[표 1] 간단한 job들의 샘플



[그림-1] 간단한 샘플의 최적화된 답

[그림-1]의 위쪽 그림을 보면 job1의 device의 각 단위시간의 상태를 나타낸 것으로서 0~1에서 켜져 있는 상태이며 작업이 진행 중인 상태이다. 그리고 1~2에서는 꺼지고 있는 상태이며 2~ 에서는 꺼져 있는 상태이다. 그러므로 0~1에서는 4, 1~2에서는 2, 2~ 에서는 각기 1로서 소비전력은 10이다. 아래쪽의 그림은 job2의 device의 상태를 나타낸 것으로 같은 방법으로 구해보면 소비전력이 21이 되어 전체 소비전력은 31이다. 이것이 *최적화*된 소비전력이라는 것이다.

data	n	T	전력소모량	Time(s)
1	2	6	31	0.09
2	3	6	47	0.04
3	6	10	174	14.36
4	8	17	321	12474.9
5	12	25	-	-
6	12	23	-	-

[표-2] 데이터들의 최적화된 소비전력

[표-2]는 각 데이터들의 최적화된 소비전력과 그 답을 구하는데 걸린 시간이다. 각 데이터들을 보았을 때 데이터의 크기가 커짐에 따라 답을 구하는 데 걸리는 시간이 아주 오래 걸린다. 하지만 소비전력은 언제나 최적화된 답을 내 놓는다는 것을 알 수 있다.

감사의 글: 본 연구는 KAIST 과학영재 교육원의 R&E 프로그램 지원을 받았음.

참고문헌

[1] V. Swaminathan et al., "Energy - Conscious, "Deterministic I/O Device Scheduling in Hard Real-Time Systems," IEEE *TCAD*, 2003.

[2] T. Kim and C. Park, "An Integrated Approach to Data Path Synthesis for Power Optimization", *J. VLSI Signal Processing,, 2000.*