충전이 가능한 전력 자원을 기반으로 한 프로세스 스케줄링 기법

송영미°, 김재훈
아주대학교 정보통신전문대학원
ymsong@dmc.ajou.ac.kr °, jaikim@ajou.ac.kr

Process Scheduling Policy Based on Rechargeable Power Resource

Young-Mi Song°, Jai-Hoon Kim
Graduate School of Information and Communication, Ajou University

요 약

지금까지 시스템의 자원을 효율적으로 활용하고, 응용 프로그램의 실시간 응답을 보장하는 등의 사용자의 요구를 만족시키기 위해 다양한 다중 프로세스 스케줄링 기법들이 연구되어 왔다. 대부분의 스케줄링 기법들은 프로세서, 즉 CPU 자원을 어떻게 할당할 것인가에 초점을 맞추고 있다. 그러나 오늘날 급격한 발전을 이루고 있는 무선 모바일 네트워크 시스템에서는 전력 자원이 네트워크의 지속시간을 결정짓는 중요한 지표가 되기 때문에, 프로세스 스케줄링 기법에 있어서도 전력 자원을 고려할 필요가 있다. 따라서 본 논문에서는 충·방전을 고려하여 전력 자원을 효율적으로 사용하기 위한 스케줄링 기법에 대해 연구하였다. 각 프로세스의 실행시간과 실행 시 소비되는 전력량을 기준으로 한 6가지의 스케줄링 기법을 제안하고 이를 비교 분석하여, 그 중 단위 실행시간 당 소비되는 전력량이 작은 순서로 프로세스를 실행시키는 스케줄링 기법이 가장 효율적임을 확인하였다.

1. 서 론

최근 중요한 이슈가 되고 있는 유비쿼터스 컴퓨팅은 센서 네트워크 및 Ad-Hoc 네트워크 기술을 바탕으로 급격한 발전을 이루어 나가고 있다. 기존의 유선 네트워크와 달리 무선 모바일 네트워크에서 각각의 노드들은 전력이나 메모리, 컴퓨팅 능력에서 매우 제한적인 자원을 가지고 있다. 특히 각 모바일 노드들은 대체가 불가능한 배터리에 의존할 수 밖에 없기 때문에, 전력 자원은 네트워크의 지속시간을 결정짓는 매우 중요한 요소이다[1]. 전력 자원을 효율적으로 사용하는 방법에 대한 기존의 연구들은 대부분 전력의 충·방전 요소는 배제하여 한정된 전력량을 가정하였고, 가능한 한 적은 전력을 사용함으로써 노드의 지속시간을 증가시키는 것에 그 초점을 맞추고 있다. 그러나 전력의 충·방전을 고려하게 되면 다른 방식의 해결방법도 가능해진다. 그 양이 고정되어 있는 CPU나 네트워크 대역폭, 메모리 등의 자원에 비해 전력 자원의 양은 시간이 지남에 따라 충전되어 증가할 수 있기 때문이다. 또한 대부분의 자원은 특정 시간에 사용하지 않게 되면 해당 자원을 낭비하는 것과 같으나, 전력 자원은 시간과 상관없이 그대로 보존된다는 특징을 가진다. 다시 말해 사용되지 않은 전력 자원은 그대로 보존되어 다른 시간에 사용할 수 있다. 이러한 특징을 가진 전력 자원을 효율적으로 사용하기 위해서는 전력을 가능한 한 적게 쓰는 것도 중요하지만, 어떤 프로세스를 먼저 실행시키는가에 따라서도 그 성능이 좌우될 수 있다. 따라서 본 논문에서는 전력 자원을 프로세스들에게 어떤 순서로 할당할 때 가장 효율적으로 사용될 수 있는지에 대해 논의하고, 시뮬레이션을 통해 단위 실행 시간당 소비되는 전력량이 작은 순서로 프로세스를 스케줄링 하는 것이 가장 효과적임을 확인하였다.

* 본 연구는 정보통신연구진흥원의 국제공동연구 지원사업, 한국과학재단의 지원사업(R01-2003-00-0794-0), 한국학술진흥재단의 지원사업(KRF-2003-003-D00375), 대학 IT 연구센터 육성•지원 산업의 연구결과로 수행되었음.

2. 관련 연구

다중 프로세스를 지원하는 운영체제에서는 일반적으로 CPU 자원을 여러 프로세스들에게 어떻게 할당할 것인가 하는 문제에 초점을 맞추고 있다. CPU자원을 기반으로 한 프로세스 스케줄링 기법으로는 FCFS, Round Robin, Shortest Process Next, Shortest Remaining Time, Highest Response Ratio Next 등이 있다[2]. 또한 실시간성을 고려한 대표적인 프로세스 스케줄링 기법으로는 Rate Monotonic, Deadline Monotonic, Earliest Deadline First, Least Slack-time First 등이 있다[3]. CPU 자원 이외에 메모리 자원에도 실시간 스케줄링을 적용하여, 페이지 교체 시에 사용되었던 기존의 LRU 기법 대신 각 프로세스의 마감 시간 또는 주기를 기준으로 페이지를 교체하는 기법이 제안되기도 하였다[4]. [5]는 센서 네트워크에서 태양 전지의 충·방전을 고려한 라우팅 기법을 제안하고 있는데, 이는 전력이 충전될 수 있는 노드를 포함하고 있는 경로를 선택함으로써 전력을 절약하는 방법이다.

3. 충·방전을 고려한 전력 자원을 기반으로 한 스케줄링 기법

단위 시간당 충전되는 전력량은 항상 일정하고, 각각의 프로세스는 일정한 실행시간과 소비 전력량을 가진다고 가정한다. 그리고 각 프로세스는 모두 동일한 우선순위를 가진다고 가정한다. 프로세스는 반드시 전체 소비 전력량을 미리 확보한 후에야 실행이 가능하며, 그렇지 않을 경우에는 필요한 양이 충전될 때까지 기다려야만 한다. 이 때 프로세스가 실행을 멈추고 필요한 전력이 충전될 때까지 기다려야만 하는 시간을 pause time이라고 정의한다. 프로세스들의 총 실행시간과 소비 전력량은 프로세스의 스케줄링 기법과 상관없이 항상 동일하다. 그러나 프로세스를 어떤 순서로 실행시키느냐에 따라 매 순간 시스템의 잔여 에너지량은 크게 달라 질 수 있다. 이는 곧 앞서 언급했던 소비되지 않으면 보존되는 전력자원의

특징 때문이다. 잔여 에너지량은 프로세스의 실행 시간이 길수록(충전 양이 많아지기 때문에), 프로세스의 소비 전력량이 적을수록 증가한다. 잔여 에너지량이 클수록 pause time은 감소할 것이므로, 효율적인 스케줄링 기법을 사용함으로써 작업의 지연시간을 감소시키고 나아가 네트워크의 지속시간을 증가시킬 수 있다. 구체적인 예를 통하여 이를 확인해 보도록 하자.

그림 1은 각 프로세스의 소비전력량을 기준으로 프로세스를 스케줄링 했을 때 각각의 잔여 에너지량과 pause time을 나타낸 것이다. 표 1은 각 프로세스의 총 소비전력량과 실행시간을 나타내고 있는데, 소비전력량이 pause time에 어떤 영향을 미치는지를 알아보기 위하여 각 프로세스는 모두 동일한 실행시간을 갖게 하였다. 초기 전력량은 100 J이고 단위시간당 충전되는 전력량은 15 J 이다.

표 1. 프로세스의 소비전력량과 실행시간

	A	B	C	D
소비전력량	80	70	5	5
실행시간	1	1	1	1

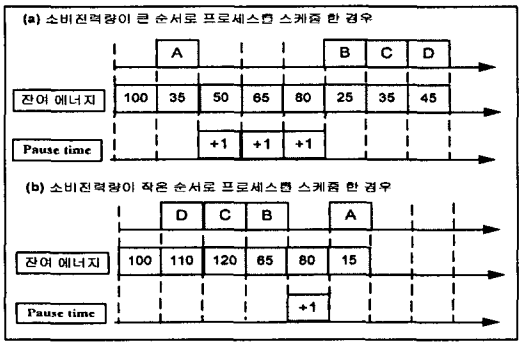

그림 1. 소비전력량에 따른 스케줄링 기법 비교

그림 1에서 보는 바와 같이 소비전력량이 작은 프로세스부터 실행시킨 경우는 소비 전력량이 큰 프로세스 순서로 스케줄링 한 경우에 비해 pause time을 2만큼 더 감소시킴을 알 수 있다. 이는 소비 전력량이 적은 프로세스를 먼저 실행시켰을 때 충전된 에너지의 많은 부분이 보존되어 잔여 에너지를 증가시키기 때문에, 그 후에 소비 전력량이 큰 프로세스를 실행시킬 때 짧은 pause time으로도 필요한 전력을 확보할 수 있게 되기 때문이다. 따라서 단위 시간당 충전되는 전력량과 소비전력량의 차가 클수록 소비전력량이 작은 프로세스부터 먼저 실행시키는 스케줄링 기법이 더욱 효과적으로 작용하게 된다.

그림2와 표2는 프로세스의 실행시간이 pause time에 어떠한 영향을 미치는지를 알아보기 위한 예이다. 이를 위해 각 프로세스의 총 소비전력량은 모두 일정한 값을 갖도록 하였고, 이때의 초기 전력량은 100 J, 단위 시간당 충전되는 전력량은 10 J 이다.

표 2. 프로세스의 소비전력량과 실행시간

	A	B	C	D
소비전력량	100	100	100	100
실행시간	1	3	7	10

그림 2는 실행시간이 긴 프로세스 순서로 스케줄링 하는 방법이 실행시간이 짧은 프로세스 순서로 스케줄링 하는 것보다 pause time을 9만큼 더 단축시킬 수 있음을 보여주고 있다. 마찬가지로, 실행시간이 긴 프로세스를 먼저 실행시키면 그만큼 충전되는 전력량도 커서 잔여 에너지를 더욱 많이 확보할

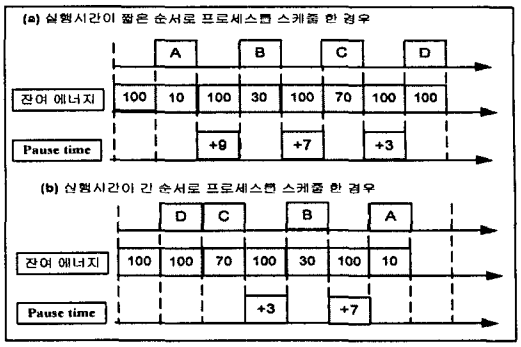

그림 2. 실행시간에 따른 스케줄링 기법 비교

수 있기 때문에, 필요한 전력을 충전하기 위한 pause time이 그만큼 감소하게 된다.

앞의 두 예를 통해, 결국 실행시간은 길고 총 소비전력량은 작은 프로세스 순서로 프로세스를 스케줄링 하는 방법이 pause time 즉, 지연시간 또는 노드의 sleep 시간을 가장 감소시키는 방법임을 알 수 있다. 실행시간이 길고 총 소비전력량이 작다는 것은 곧 단위시간당 소비전력량이 작음을 의미한다. 다음절에서는 프로세스의 실행시간과 소비전력량을 기준으로 하는 6가지의 스케줄링 기법을 제안하고, 시뮬레이션을 통해 각각의 성능을 비교해 보았다. 그 결과 단위 시간당 소비전력량이 작은 프로세스 순서로 스케줄링 하는 방법이 가장 효율적임을 확인하였다.

4. 시뮬레이션 및 성능 평가

프로세스는 실행시간과 소비전력량에 따라 다음과 같은 6가지의 스케줄링 방법으로 실행될 수 있다.

1. 전체 소비 전력량이 큰 프로세스부터 실행시키는 방법
2. 전체 소비 전력량이 작은 프로세스부터 실행시키는 방법
3. 프로세스의 실행시간이 긴 프로세스부터 실행시키는 방법
4. 프로세스의 실행시간이 짧은 프로세스부터 실행시키는 방법
5. 단위 시간당 소비 전력이 큰 프로세스부터 실행시키는 방법
6. 단위 시간당 소비 전력량이 작은 프로세스부터 실행시키는 방법

6가지의 스케줄링 방법과 함께 고려해 볼 것은 시뮬레이션 시 어떤 프로세스를 대상으로 할 것인가 하는 점이다. 응용프로그램에 따라 실행시간이 짧으면 사용하는 전력량도 작고, 실행시간이 길면 사용하는 전력량도 큰 프로세스가 존재할 것이다. 반면 실행시간이 짧더라도 많은 디바이스를 동시에 사용하기 때문에 소비 전력량이 큰 프로세스도 있고, 실행시간이 길지만 대부분의 시간을 대기 상태에 있어서 소비 전력량이 작은 프로세스도 존재할 것이다.

실행시간과 소비 전력이 비례하는 프로세스들을 대상으로 하여 6가지의 스케줄링 기법을 적용해본 시뮬레이션의 결과는 그림 3과 같다. 소비전력량이 80~100 J인 프로세스는 8~10의 단위 실행시간 값을 갖고, 소비전력량이 10~30 J인 프로세스는 1~3의 단위 실행시간 값을 갖도록 하였다. 그 결과, 위의 6가지 스케줄링 방법 중 1,3,5번보다 2,4,6번의 방법이 더욱 효율적인 것으로 나타났다. 가장 효율적인 것은 단위 시간당 소비 전력이 작은 순서로 스케줄링 하는 방법이며, 가장 비효율적인 단위 시간당 소비 전력이 큰 순서로 스케줄링 하는 방법보다 pause time이 최대 96.2% 정도 감소되었다. 반면, 그림 4는 실행시간과 소비전력이 반비례하는 프로세스들을 대상으로 하여 시뮬레이션 한 결과이다. 이 시뮬레이션에서는 소비전력량이 80~100 J인 프로세스는 1~3의 단위 실행

시간 값을 갖고, 소비전력량이 10~30 J인 프로세스는 8~10의 단위 실행시간 값을 갖게 하였다. 그림 4에서는 그림 3과 달리 3번 방법이 4번 방법보다 훨씬 효율적이다. 이는 그림 3의 시뮬레이션에서는 실행 시간이 짧은 프로세스의 단위 시간당 소비 전력량이 실행시간이 긴 프로세스보다 작았으나, 그림 4의 시뮬레이션에서는 더 크기 때문이다. 결국, 단위 시간당 소비 전력량이 작은 순서로 스케줄링 하는 방법이 가장 효과적임을 알 수 있다. 그림 4에서도 마찬가지로 pause time이 최대 92.6% 정도 감소되었다. 또한 그림3에 비해 그림 4의 pause time이 매우 큼을 알 수 있는데, 이는 그림 4의 시뮬레이션이 그림 3보다 실행시간에 비해 소비 전력량이 매우 큰 프로세스들을 많이 가지고 있어 pause time을 크게 증가시키기 때문이다.

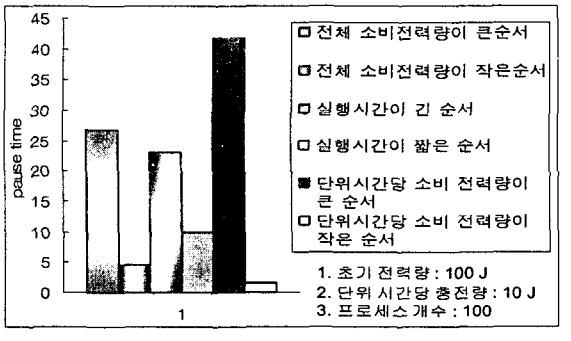

그림 3. 실행시간과 소비전력량이 비례하는 프로세스를 대상으로 한 스케줄링 기법의 성능 비교

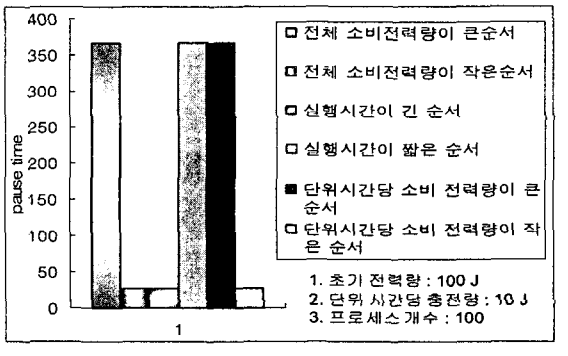

그림 4. 실행시간과 소비전력량이 반비례하는 프로세스를 대상으로 한 스케줄링 기법의 성능 비교

그림 5는 프로세스의 개수가 증가함에 따라 스케줄링 기법의 성능을 비교한 그래프이다. 프로세스의 실행 시간과 소비 전력량은 각각 1~10, 10~100사이의 값 중에서 랜덤하게 선택하였고 이때의 단위 시간당 충전량은 10 J이다. 이 경우에도 마찬가지로 단위 시간당 소비 전력량이 큰 순서로 스케줄링 하는 방법이 가장 효율적이다. 또한 프로세스의 개수가 증가할수록 그 성능차이가 더욱 뚜렷하다.

그림 6은 그림 5와 같은 환경에서 단위 시간당 충전되는 전력량을 변화시켜 가면서 성능을 측정해 본 결과이다. 단위 시간당 충전되는 전력량이 작을 때에는 각 스케줄링 방법 마다 그 성능이 큰 차이를 보이고 있지 않으나, 단위 시간당 충전되는 전력량이 커질수록 최대 72%정도 pause time이 감소하였다.

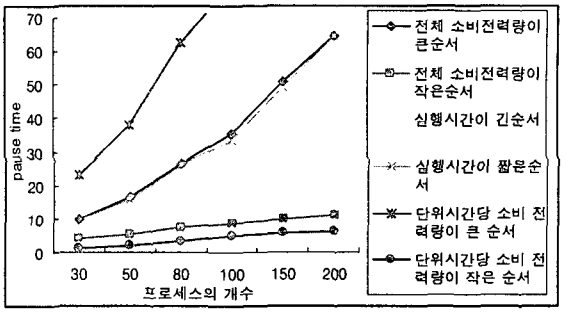

그림 5. 프로세스의 개수에 따른 스케줄링 기법의 성능 비교

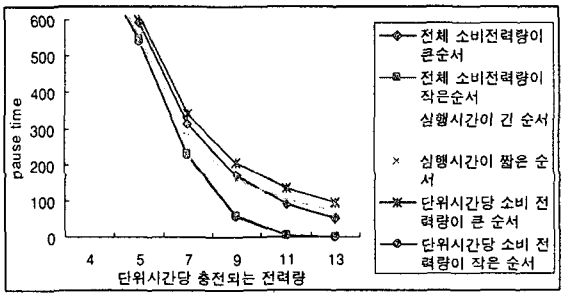

그림 6. 단위 시간당 충전되는 전력량에 따른 스케줄링 기법의 성능 비교

5. 결론 및 향후 과제

본 논문에서는 충전이 가능한 전력 자원을 기반으로 한 프로세스 스케줄링 기법에 대해 연구하여, 프로세스의 실행 순서만 바꾸어도 전력 자원을 훨씬 더 효율적으로 사용할 수 있음을 확인 하였다. 각 프로세스의 실행시간과 소비전력량에 따라 6가지 스케줄링 기법을 제안하고 시뮬레이션을 통해 성능을 측정하고 분석한 결과, 단위 시간당 소비 전력량이 작은 프로세스 순서로 스케줄링 하는 방법이 가장 효율적임을 확인하였다. pause time 즉 전력이 방전되어 다시 충전될 때까지 지연 되는 시간을 성능 평가 요소로 하여, 최대 96% 정도의 pause time 감소가 가능함을 증명하였다. 이러한 지연 시간의 감소는 다중 멀티 프로세스 환경의 일 처리 능률을 향상시키고 QoS를 증가시키며, 무선 모바일 환경에서 노드의 sleep 시간을 줄이고 전체 네트워크의 지속 시간을 향상시키는 역할을 한다. 향후 연구로는 충·방전을 고려한 전력 자원을 라우팅 기법에도 적용시켜 보는 것과, 충전되는 전력량이 일정하지 않은 환경에서의 프로세스 스케줄링 기법에 대한 것 등이 있다.

[참 고 문 헌]

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci "Survey on Sensor Networks," *IEEE Communications Magazine*, Aug. 2002.

[2] William Stallings, "Operating Systems, Internals and Design Principles", Prentice Hall, 2001.

[3] Maurice J. Bach, "Design of the Unix Operating System", Prentice Hall, 1987.

[4] 가진호, "실시간 시스템을 위한 실시간 메모리 교체 기법", *한국정보과학회, 2001*

[5] Thiemo Voigt, Hartmut Ritter and Jochen Schiller, "Solar-aware Routing in Wireless Sensor Networks", PWC, 2003.