모바일 장치를 위한 사용자 입력 및 프로세스 정보에 기반한 전력 관리 기법

김영훈 박대준 신동군
성균관대학교 컴퓨터공학과
ehdeoddl@skku.edu, pdaejun@skku.edu, dongkun@skku.edu

User Input and Process-Based Power Management Technique for Mobile Devices

Younghun Kim, Daejun Park, Dongkun Shin
Deparment of Electrical and Computer Engineering, Sungkyunkwan University

요 약

안드로이드 기반의 스마트폰은 소비전력을 줄이기 위해 DVFS와 거버너를 사용한다. 그러나 일반적으로 사용되는 온디맨드 거버너는 반응속도가 떨어지고, 모바일 환경의 특징을 제대로 반영하지 못한다는 문제점이 있다. 본 논문에서는 온디맨드 거버너가 가지고 있던 문제점을 개선하고 모바일 환경의 특징을 반영한 온액션(Onaction) 거버너를 제안한다. 온액션 거버너는 온디맨드 거버너를 기반으로 가중치를 사용하여 클럭을 더 동적으로 조절한다. 실험 결과 온액션 거버너는 온디맨드 거버너에 비해 소비전력과 반응속도, 초기 로딩 속도 모두 더 좋은 성능을 보였다.

1. 서 론

스마트폰과 같은 모바일 기기의 비중이 커지면서 해당 기기의 사용 가능 시간이 중요한 이슈가 되었다. 모바일 기기의 사용가능 시간은 소비전력과 배터리 용량으로 결정이 된다. 따라서 사용시간을 늘리기 위해서는 배터리 용량을 늘리거나 소비전력을 줄여야 한다. 하지만 배터리 용량을 늘리는 것은 기술적인 한계가 있으므로, 일반적으로 소비전력을 줄여 사용시간을 늘린다.

안드로이드 기반 스마트폰에서 소비전력을 줄이기 위해 사용되는 대표적인 기술이 DVFS(Dynamic Voltage and Frequency Scailing)와 거버너(governor)이다. DVFS는 하드웨어 적으로 CPU의 전압과 주파수를 동적으로 바꾸어 소비전력을 줄인다. 거버너는 커널상에서 DVFS를 어떻게 컨트롤 할지를 결정하는 역할을 수행한다. 거버너는 여러 종류가 있지만 일반적으로 CPU 사용율(Utilization)을 기준으로 하는 온디맨드 (ondemand) 거버너가 사용된다.

그러나 온디맨드 거버너는 DVFS를 사용하지 않을 때에 비해 반응성이 떨어진다는 단점이 있다. 또한 해당 거버너가 데스크탑 기반 환경에서 만들어졌기 때문에 아래와 같은 모바일 환경의 특징은 고려되어 있지 않다.

안드로이드 기반의 스마트폰과 같은 모바일 환경은 데스크탑 환경에 비해 전체 작동중인 프로세스의 수가 적으며, 한번에 하나의 프로그램만이 사용자와 상호작용을 수행한다. 두 번째로 사용자의 입력 수단과 그에 따른 모바일 기기의 반응이 한정적이다. 따라서 모바일 기기와 사용자간의 상호작용을 몇 가지 단계로 구분 가능하다

본 연구는 지식경제부 및 한국산업기술평가관리원의 산업융합원천기술개발사업(정보통신)의 일환으로 수행하였음.
[KI0018-10041244, 스마트TV 2.0 소프트웨어 플랫폼]

본 논문에서는 온디맨드 거버너의 단점이었던 반응속도 문제를 해결하고, 모바일 환경의 두 가지 특성을 모두 고려한 새로운 거버너인 온액션(Onaction) 거버너를 제안한다. 온액션 거버너의 구현에 대한 상세한 내용을 설명하고 실험 결과를 통해 온디맨드 거버너에 비해 소비전력이나 성능 면에서 어느 정도의 향상이 있는지를 확인하겠다.

2. 관련 연구

먼저 모바일 기기에서 DVFS를 이용한 소비전력 절감에 한계가 있다는 논의가 있었다.[1] 모바일 기기 전체에서 CPU가 소비하는 전력이 낮기 때문에 DVFS를 통한 소비전력 절감은 한계가 있다는 내용이다. 하지만 최근에 사용되는 ARM A15 프로세서의 경우 최대 소비전력이 7W 이상으로[2] CPU 소비전력이 크게 증가하였다. 따라서 현재는 DVFS로 소비전력 절감 효과를 얻는 것이 가능하다.

다음으로 앞서 언급한 모바일 환경의 특성 중 하나를 반영한 거버너가 제시된 적이 있다.[3] 그러나 사용자 입력에만 기반하여 클럭을 조정하였기 때문에, 프로세스의 정보가 활용되지 않았다. 또한 거버너에 대한 성능 평가를 블라인드 테스트를 통해 성능에 큰 차이가 없었다는 식으로 결론을 내려 연구결과에 대한 신뢰성이 부족하였다.

3. 온디맨드 거버너 (Ondemand Governor)

온디맨드 거버너는 샘플링 주기(sampling rate)마다 거버너가 호출되어 CPU 사용율을 기준으로 클럭 변경을 결정한다. 예를 들어 사용자가 기기를 사용하지 않을 때는 CPU 사용율은 낮아지고 클럭을 낮게 유지하여 소비전력을 절감한다. 반대로 사용자가 기기를 사용하여 CPU 사용율이 상승하면 거버너는 클럭을 높여 처리속도를 향상시킨다.

온디맨드 거버너는 사용자가 어떤 일을 요청하였을 때, CPU 사용율이 일정 값까지 상승하기 전까지는 낮은 클럭으로

동작한다. 따라서 DVFS를 사용하지 않을 때에 비해 상대적으로 반응 속도가 낮아진다는 단점이 있다.

4. 온액션 거버너(Onaction Governor)의 구현

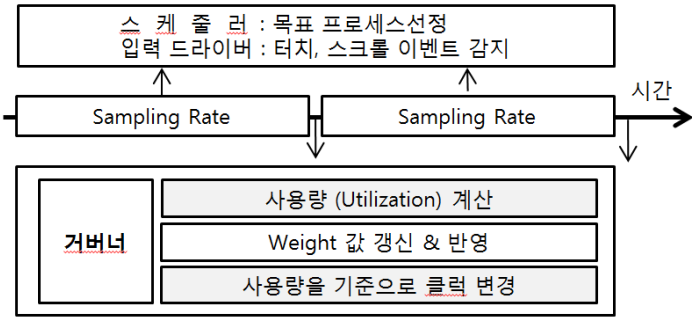


그림 1 거버너의 구조

이 장에서는 새로 제시한 온액션 거버너를 어떻게 구현하였고, CPU 사용율에 적용되는 가중치 값을 어떻게 처리하는지에 대해 설명한다.

온액션 거버너는 온디맨드 거버너에서 CPU 사용율에 가중치 (weight)를 주는 형태로 구현하였다. 가중치는 여러 프로세스에 대해 서로 다르게 동작하기 위해, 프로세스 정보를 가지는 task_struct 구조체에 개별적으로 저장된다.

[그림 1] 은 시간 흐름에 따라 클럭이 어떻게 변경되는지를 나타낸 것이다. 먼저 샘플링 기간 동안 거버너에서 사용할 작동중인 프로세스에 대한 정보와 터치 관련 정보를 처리한다. 거버너가 실행되면 첫 번째로 가중치가 적용되지 않은 CPU 사용율을 계산한다. 두 번째로 터치나 스크롤 입력여부에 따라 네 가지 상호작용 단계 중 하나를 선택한다. 그리고 스케줄러를 통해 선택된 프로세스가 가지고 있는 가중치들 중 해당 단계에 대응하는 가중치를 갱신한 뒤 CPU 사용율에 반영한다. 마지막으로 거버너는 가중치가 적용된 CPU 사용율을 기준으로 클럭을 변경한다. 이 과정 중에서 첫 번째와 세 번째 과정은 기존 온디맨드 거버너와 동일하며, 새로 추가된 두 번째 부분에 대해서 자세히 설명한다.

4.1 상호작용 단계와 가중치 값

가중치는 사용자와 모바일 기기간의 상호작용을 몇 가지 단계로 설정하고 단 단계별로 개별적인 값을 가지도록 하였다. 상호작용 단계는 간략하게 몇 가지 단계로 나눌 수 있는데 [3] 그 중 사용자 입력을 터치와 스크롤로 구분하여 4가지 단계로 설정하였다. 터치와 스크롤을 구분한 이유는 각각의 사용자 입력에 대한 반응을 세분화 하기 위해서이다.

4.1.1 어플리케이션 시작 단계

프로세스를 생성하고 어플리케이션(Application)이 시작되는 단계로 해당 프로세스의 시작시작과 시스템의 현재 시간을 비교하여 판단한다. 이 단계에서는 어플리케이션의 로딩시간을 줄이기 위해서 고정된 가중치를 반영하여 평균 클럭을 높게 유지한다.

4.1.2 터치 반응 단계

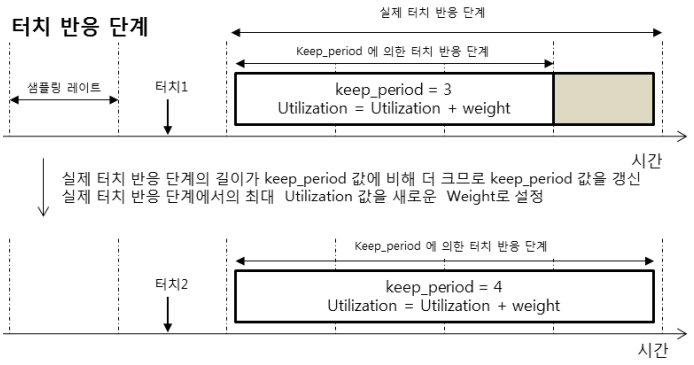
이 단계는 샘플링 기간 동안 발생한 터치 이벤트에 반응하는 단계이다. 이 단계에서는 가중치의 크기 이외에도, 해당 가중치를 얼마나 유지 할지가 중요한 요소가 된다. 이를 위해 keep_period 라는 변수를 사용하여 몇 번의 샘플링 기간 동안 가중치를 유지할 것인지를 결정한다. [그림 2-a]는 가중치와 keep_period값이 어떻게 갱신 되는지를 나타낸 것이다. 터치 입력 후 CPU 사용율이 일정 값 이하로 떨어질 때까지의 구간을 실제 터치 반응 단계로 파악한다. 그리고 해당 구간의 길이를 keep_period와 비교한 뒤 keep_period값을 실제 터치 반응 단계 구간의 길이에 가깝게 갱신한다. 가중치는 해당 구간동안의 최대 CPU 사용율로 갱신한다.

4.1.3 스크롤 반응 단계

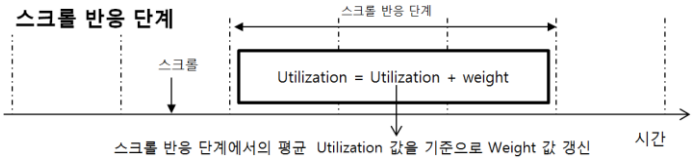
샘플링 기간 동안 스크롤 이벤트에 대응하는 단계이다. [그림 2-b]와 같이 스크롤 반응 단계 동안의 평균 CPU 사용율을 계산한 뒤 이전 스크롤 반응 단계 시와 비교하여 가중치를 갱신한다 현재 반응 단계의 평균 CPU 사용율이 높은 경우 가중치를 높여 스크롤 반응 단계에서 클럭 상승이 더 일어 날 수 있도록 하며, 반대의 경우 가중치를 줄인다.

4.1.4 대기 단계

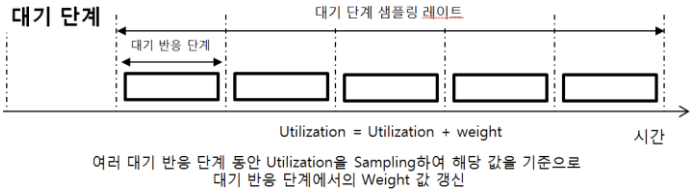
[그림 2-c]와 같이 가중치는 여러 대기 단계의 CPU 사용율을 샘플링 하여 이전 단계에서 샘플링 한 값과 비교하여 높이거나 낮추는 형태로 갱신한다. 대기 단계에서의 CPU 사용율이 높은 경우 가중치는 최대 0이 되어 온디맨드 거버너와 동일하게 동작한다. 반대의 경우 가중치는 - 값을 가지며 클럭을 더 낮추어 소비전력을 줄인다.



(a) 터치 반응 단계



(b) 스크롤 반응 단계



(c) 대기 단계

그림 2 반응 단계 별 가중치 갱신

5. 실험 설계

이 장에서는 실험 항목으로 설정한 요소들과 해당 항목들의 측정을 위해 실험 설계를 어떻게 하였는지 이야기 한다.

5.1 소비전력

표 1소비전력 계산 식과 각 계수

Clock (MHz)	Idle (Wh)	Full Load (Wh)
250	0.559	0.710
300	0.559	0.774
350	0.602	0.839
400	0.602	0.882
450	0.602	0.946
500	0.602	1.054
550	0.602	1.161
600	0.645	1.290

Clock (MHz)	Idle (Wh)	Full Load (Wh)
800	0.645	2.333
900	0.645	2.623
1000	0.688	3.096
1100	0.688	3.849
1200	0.688	4.730
1300	0.688	5.547
1400	0.688	6.579
1500	0.688	6.891
1600	0.688	7.891

$$소비전력 = (\text{Full Load} - \text{Idle})_{clock} * Utilization + Idle_{Clock} \quad (1)$$

온액션 거버너가 온디맨드 거버너에 비해 소비전력이 얼마나 감소 하였는지 비교하기 위해 선택했다.

[표 1]에서 Full Load는 CPU 사용율이 100% 일 때 각 클럭 별로 소비전력을 측정한 값이다. 측정과정에서 wifi나 블루투스등 부가기능과 디스플레이, 통신기능을 끈 상태에서 측정하였다. 또한 GPU도 거의 사용하지 않도록 하여, 기기 작동을 위한 최저한의 전력과 CPU가 소비하는 전력을 포함한 값을 구하였다. 이를 통해 거버너가 호출 될 때 마다 CPU 사용율과 클럭값을 기록한 뒤 위의 공식(1)을 이용해 소비전력을 계산하였다. [4, 5]

5.2 반응속도

온디맨드 거버너의 낮은 반응성 문제를 해결하였는지 확인하기 위해 선택하였다. 측정은 인풋 드라이버와 비디오 드라이버를 이용하여, 사용자가 터치한 시점에서부터 비디오 드라이버의 프레임 버퍼에 새로운 화면을 그리는 요청이 들어갈 때까지의 시간 간격을 측정했다.

5.3 초기 로딩 시간

어플리케이션 시작 단계가 제대로 작동하는지를 확인하기 위해 선택했다. 터치 이벤트를 발생시켜 어플리케이션을 실행시키는 시점에서부터 로딩이 끝나 프레임 버퍼상에 새로운 요청이 들어올 때까지의 시간 간격을 측정했다.

6. 실험 결과

대상 어플리케이션 플레이 스토어의 인기 무료 리스트에서 임의로 선택하였다. 선택한 어플리케이션은 “Angry Bird Star Wars” 와 “Dady was a thief” 이며, 실험은 갤럭시 S4에서 수행하였다. 실험에 대한 각 결과값은 아래의 [표 2]와 같다.

6.1 소비전력

3분간 총 10회 수행하여 평균 소비전력을 구하였다. 실험 결과 소비전력이 각각 15%, 7% 감소하였다. 이는 상호작용 단계 중 대기 단계에서 온디맨드 거버너에 비해 클럭을 더 낮추면서 전체 소비전력이 감소했기 때문이다.

6.2 반응 속도

메뉴전환 속도를 20회씩 측정하여 평균값을 계산하였다.

- 앵그리 버드 : 혹성 선택 메뉴에서 메인 메뉴로 전환
- Dady was a thief : 메뉴에서 언어 선택 화면으로 전환

실험 결과 메뉴 전환 속도가 각각 15%, 11% 상승했다. 이는 온액션 거버너가 사용자 입력을 감지하였을 때 CPU 사용율에 높은 가중치를 적용하여 온디맨드 거버너에 비해 더 빠르게 높은 클럭으로 변경되기 때문이다.

6.3 초기로딩

각각 10회씩 측정하여 평균값을 계산하였다. 실험 결과 초기 로딩 속도가 각각 8%, 11% 빨라짐을 확인하였다. 이는 앱 시작 단계에서 고정된 가중치를 적용하여 온디맨드 거버너보다 평균 클럭이 높아졌기 때문이다.

표 2 실험 결과

실험 항목		AngryBird	Dady
소비전력 (mW)	온디맨드	48.8	45.1
	온액션	41.4	42.1
	감소율	**15%**	**7%**
반응시간 (ms)	온디맨드	314	67
	온액션	266	60
	감소율	**15%**	**11%**
로딩시간 (s)	온디맨드	6.07	6.91
	온액션	5.58	6.17
	감소율	**8%**	**11%**

7 결론

기존에 사용되던 온디맨드 거버너는 반응속도가 떨어지고, 모바일 환경의 특성을 반영하지 못한다. 본 논문에서는 온디벤드 거버너를 기반으로 하여 모바일 환경을 반영하고 반응속도를 개선한 새로운 거버너를 제시했다. 새로 제시한 온액션 거버너는 상호작용 단계와 프로세스 정보에 따라 CPU 사용율에 서로 다른 가중치를 반영한다. 따라서 온디맨드 거버너보다 동적으로 클럭을 변경하여 성능과 소비전력 면에서 더 이득을 얻을 수 있다. 그리고 로딩 속도, 반응 속도, 소비전력 각 부분에 대한 실험결과를 통해 평균적으로 10% 정도의 향상이 있음을 확인하였다.

추가적으로 제조사에서 기본 탑재된 애플리케이션에 대해 거버너에서 사용하는 가중치와 keep_period의 최적값을 미리 설정하여 성능을 높이고 소비전력을 줄이는 것도 가능하다.

참고문헌

[1] 홍성수. DVFS 에 기반한 안드로이드 스마트폰의 CPU 소모 전력 절감 기법의 한계. 정보과학회지, 30.7, 9-16, 2012.

[2] Mathieu, In Kernel Switcher:A solution to support ARM's new big.LITTLE technology, Embedded Linux Conference, 2013

[3] 우상정, et al. 사용자 입력 기반의 스마트폰 전력 감소 기법. 정보과학회논문지: 시스템 및 이론, 40.1, 45-51, 2013.

[4] JUNG, Wonwoo, et al. DevScope: a nonintrusive and online power analysis tool for smartphone hardware components, CODES+ISSS, 353-362, 2012.

[5] YOON, Chanmin, et al. Appscope: Application energy metering framework for android smartphone using kernel activity monitoring,USENIX ATC, 2012.