안드로이드 환경에서의 에너지 절약을 위한 효율적인 CPU 자원 활용 기법

류준한* · 권영호* · 이병호*

*한양대학교

Efficient CPU Resource Utilization Mechanism on Android Platforms for Conserving Energy

Jun-han Ryu* · Young-ho Kwon* · Byung-ho Rhee**

*Hanyang University

E-mail : juju1013@hanyang.ac.kr

요 약

스마트폰 산업이 발전하면서 내부 하드웨어 장치들이 고사양의 장치가 되었고 이전 보다 많은 전력소비를 요구 한다. 그러므로 고용량의 배터리가 필요 하지만 스마트폰의 소형화로 인해 부피가 큰 배터리를 탑재하는 것에 한계가 있다. 리눅스 커널은 이러한 한계점을 소프트웨어 기술로 보완하기 위해 DVFS Mechanism을 제공 한다. DVFS는 동적으로 CPU의 동작주파수를 조정 하여 CPU의 전력소비를 줄이는 Mechanism이다. DVFS에서 기본정책인 ondemand는 up_threshold를 넘을 때 마다 최대 동작주파수를 적용 하여 상당 시간 유지되므로 CPU 자원의 낭비를 초래 한다. 본 논문에서는 이러한 점에 착안하여 계속해서 현재 동작주파수 대비 높은 CPU 이용률을 유지함으로써 CPU자원의 낭비를 막고 에너지를 절약 하는 기법을 제안한다.

ABSTRACT

as the smartphone industry developed, the smartphone's internal hardware devices have become high-end devices and it requires more power consumption than the previous one. therefore a battery of high capacity needed, but there is a limit in order to equip a large battery on account of smartphone minimization. The Linux Kernel provides the DVFS Mechanism to compensate for these limitations by software techniques. DVFS is dynamically adjust the frequency of the CPU to reduce the power consumption of the CPU. ondemand governor, the default policy in DVFS, apply the maximum frequency of the CPU whenever exceeding the up_threshold. so it result in a waste of CPU resources. by paying attention to this point, this paper propose the mechanism that maintain a high CPU utilization in proportion to the current frequency of the cpu to prevent the waste of CPU resources and conserve energy.

키워드

battery energy consumption cpu utilization

Ⅰ. 서 론

스마트폰 산업은 현재 성장기를 지나 성숙기에 접어들었다. 스마트폰의 성능 또한 비약적으로 상승했다. 고사양의 하드웨어 장치들은 이전장치들보다 더욱 많은 전력소비를 요구한다. 스마트폰의 사양은 점점 더 향상 되고 장비의 크기는 줄어들어 장비에 탑재되는 하드웨어들의 소형화도 수반되었다. 그러나 다른 하드웨어 장치들에 비해 배터리산업의 기술 성장이 느렸기 때문에 고 용량의 배터리를 소형화시키는 것에 한계가 있었다. 따라서 정해진 배터리 용량 안에서 배터리의 수명을 연장시키기 위한 방법으로 소프트웨어 기술을 이용한 방법이 요구된다.

스마트폰의 전체 소비전력에서 CPU 장치의 소비전력[1]은 12%~13%로 매우 높은 비중을 차지한다. 그러므로 CPU의 전력소비를 줄이기 위해서 Linux Kernel은 CPU 동작주파수를 동적으로 변경할 수 있는 DVFS(dynamic voltage frequency scaling) 기법[2]을 제공 한다. DVFS는 CPU 동작주파수를 변경하는 모듈인 governor를 이용한다. governor의 종류는 보통 performance, powersave, ondemand, userspace가 있다. performance는 CPU 동작주파수를 최대치로 유지하는 정책이며, powersave는 CPU 동작주파수를 최저치로 유지하는 정책이다. ondemand governor는 CPU 이용률이 정해 놓은 up_threshold를 넘을 시에는 동작주파수를 최대치로 변경하고, CPU 이용률이 down_threshold보다 적으면 동작주파수의 20%씩 하향 조정하는 정책이다. 하지만 ondemand 방식은 동작주파수의 최대치가 필요하지 않은 상황에서도 동작주파수를 최대치로 조정함으로써 실제 필요한 동작주파수와 최대 동작주파수와의 차이만큼 전력 낭비가 발생하게 된다. 불필요하게 높은 동작주파수가 유지 되는 동안 CPU 이용률이 낮아지고 불필요한 전력 소비가 발생하게 된다. 본 논문에서는 이 점에 착안하여 CPU 이용률을 profiling하고 CPU의 이용률이 일정하게 높은 비율을 유지 하도록 하여 CPU의 자원 낭비를 막음으로써 소비전력을 줄이는 방법을 제안한다.

Ⅱ. 본문

그림 1은 제안된 governor인 RRCU(Regularated cpu usage) governor의 architecture이다. 필요한 모든 데이터는 kernel의 system file을 통해 수집한다. 그리고 system file의 정보를 수집하는 profiler는 android service로 구현되어 있다. android에 구현된 profiler는 kernel의 system file proc/stat에서 cpu 이용률을 수집한다. cpu 동작주파수 정보는 DVFS의 cpuinfo_cur_freq로부터 현재 cpu frequency를 수집한다.

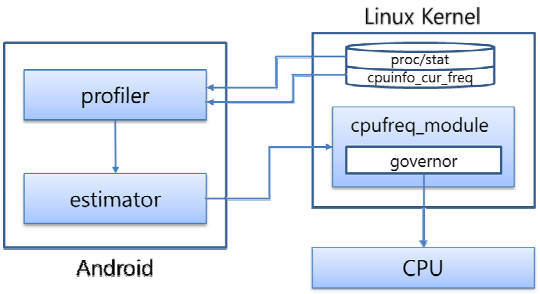

그림 1. RRCU governor architecture

```
begin
if cur_cpu_load more than up_threshold then
   begin
      if cur_cpu_frequency equal max_frequency then
         begin
            sampling_rate delay, continue
         end
      else
         begin
            search for target_frequency using
equation(3)
         end
   end
else if current_cpu_utilization less than down_threshold
then
   begin
      if current_cpu_frequency equal min_frequency then
         begin
            continue
         end
      else
         begin
            search for target_frequency using
equation(3)
         end
   end
else
   begin
      continue
   end
end
```

그림 2. 제안된 알고리즘

수집한 데이터를 이용하여 estimator는 cpu 이용률이 60% 이상이 되도록 하는 cpu frequency 값을 계산하고 해당 값을 kernel의 cpufreq module에 전달하여 userspace governor를 통해 CPU의 동작주파수를 변경한다.

그림 2는 RRCU의 알고리즘이다. cpu 이용률이 up_threshold 보다 높지만 현재 cpu 동작주파수가 최대치이면 profiling, estimator과정을 거치지 않음으로써 overhead를 줄이며 cpu 동작주파수가 최대치가 아닐 경우에는 profiling, estimator 과정을 통해 cpu 이용률이 60%~80%가 되도록 수식 (3)을 이용하여 목표 cpu 동작주파수를 계산 후에 적용한다. 만약 cpu 이용률이 down_threshold 보다 낮고 현재 동작주파수가 최저치이면 profiling, estimator과정을 생략하여 overhead를 줄이고 그렇지 않으면 profiling, estimator과정을 통해 cpu 이용률이 60%~80%가 되도록 수식 (3)을 이용하여 목표 cpu 동작주파수를 계산 후에 적용한다.

$$cpu\,rate_{utilization} = \frac{User_{diff} + Nice_{diff} + System_{diff}}{User_{diff} + Nice_{diff} + System_{diff} + Idle_{diff}} \times 100$$

수식 1. cpu 사용률을 구하는 식

proc/stat은 cpu 이용률을 jiffies의 크기로 나타내고 각 필드는 순서대로 user mode, nice mode, system mode, idle mode로 구성 되어 있다. 수식

(1)은 각 필드를 sampling_rate 간격으로 수집한 값의 차이를 이용 하여 cpu 이용률을 구하는 식이다.

본 논문에서는 sampling_rate를 40ms 로 고정한다. 그 이유는 다양한 sampling_rate 값을 실험한 결과 만약 sampling rate가 40ms보다 크다면 cpu 동작주파수를 변경하는 시간이 길어져 사용자 입력에 대한 반응성이 낮아졌고 반대로 40ms보다 작으면 profiling과 estimating 이 잦아지면서 overhead가 커지게 되어 cpu 이용률이 불필요하게 증가하여 전력소비 감소 효과가 줄어드는 결과를 가져왔기 때문이다.

$$cpu\,frequency_{used} = cpu\,frequency_{current} \times \frac{cpu\,rate_{utilization}}{100}$$

수식 2. cpu 사용률을 동작주파수로 표현하는 식

$$cpu\,frequency_{target} = \frac{cpu\,frequency_{used}}{cpu\,rate_{utilization}} \times cpu\,rate_{target}$$

수식 3. 정해진 cpu 사용률을 위한 cpu 동작주파수를 구하는 식

profiling을 마치면 estimator에서 cpu utilization이 60% 이상이 되도록 하는 cpu frequency를 계산하게 된다. 현재 cpu 이용률을 통해 현재 cpu frequency에서 실제 필요한 cpu frequency 값을 계산한다.

수식 (2)는 profiler에서 수집된 데이터를 통해 현재 cpu 동작주파수에서 cpu 이용률이 차지하는 동작주파수의 크기를 계산하는 수식이다. 현재 cpu 동작주파수의 크기를 구하고 나면 수식 (3)을 통해 cpu 이용률이 60%가 되는 target frequency를 계산한다. target frequency를 정한 후에 마지막으로 DVFS userspace governor를 통해 target frequency를 적용한다.

Ⅲ. 실험 및 평가

실험은 베가레이서(IM-A760S)기기에서 안드로이드 Jelly Bean(4.1), Kernel(3.0.31) 환경에서 실시하였다. 전류측정은 Arduino Uno와 INA219 전류센서를 이용 하여 측정 하였다. cpu 이용률 변화 추이를 측정하기 위해 3분 50초 크기의 동영상을 플레이 하여 측정했으며 전류측정을 위해 벤치마크용 애플리케이션을 작성하여 측정하였다. 사용한 애플리케이션은 비디오플레이, 실수연산, 정수연산, I/O연산을 순차적으로 처리한다.

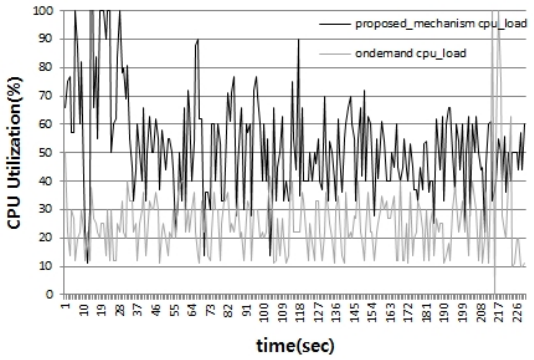

그림 3. CPU 이용률 변화 추이 그래프

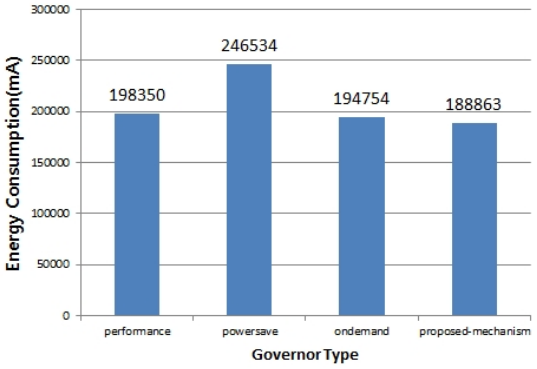

그림 4. governor 별 전류 측정 그래프

그림 3은 3분 50초 크기의 동영상을 플레이 할 때의 cpu 이용률 변화이다. ondemand governor는 10%에서 30% 값에 몰려있으며 제안된 알고리즘인 RRCU governor는 40%에서 70% 값에 몰려 있는 것을 확인 할 수 있다. 이것은 최초 설정한 목표인 60%대의 cpu 이용률을 유지하는 것을 확인할 수 있는 실험결과이다.

그림 4는 벤치마크용 애플리케이션을 수행 했을 때의 각 governor 별 전류 소모량을 측정한 것이다. 제안한 알고리즘인 RRCU governor는 네 개의 governor중 가장 낮은 전류 소모량을 보였다. 가장 낮은 동작주파수를 사용하는 powersave는 가장 낮은 전류소모량을 예측 하였으나 프로그램 수행시간이 길어지면서 오히려 가장 높은 전류소모량을 보였다.

Ⅳ. 결 론

본 논문에서는 ondemand 방식의 cpu 사용에 대한 비효율성에 주목하여 항상 높은 cpu 이용률을 유지하여 cpu의 효율성을 높여 에너지 소비를 줄이는 방식을 제안했다. 실험을 위해 일정한 cpu

이용률을 보이는 동영상 재생과 벤치마크를 위한 애플리케이션을 작성하여 실험을 했고 실험 결과 ondemand 방식 보다 제안한 알고리즘이 4% 전력 절감 효과가 있음을 보였다. 본고에서는 CPU 이용률만을 고려하였지만 CPU이용률 외에 프로세스 별 timeslice를 고려하여 CPU사용에 대한 예측을 한다면 더 높은 효율을 보일 것으로 예상한다.

참고문헌

[1] 유종훈・허승주・홍성수, “DVFS에 기반한 안드로이드 스마트폰의 cpu 소모 전력 절감 기법의 한계”, 정보과학학회지, 제30권 제7호, 9-16(8page), 2012.7
[2] DVFS User Guide, http://processors.wiki.ti.com/index.php/DVFS_User_Guide
[3] 우상정・서원익・김창대・허재혁, “사용자 입력 기반의 스마트폰 전력 감소 기법”,한국정보과학회 2012한국컴퓨터종합학술대회 논문집, 제39권 제1호, 4-6(3pages), 2012.6
[4] Wen-Ywew Liang・Po-Ting Lai・Che-Wun Chiou, “An Energy Conservation DVFS Algorithm for the Android Operating System”, 한국정보기술융합회 제1권, 93-100(8pages), 2010.12
[5] Linux Kernel DVFS API, http://trac.gateworks.com/wiki/DVFS