열 평형 상태를 이용한 프로세서 전력 모델 계산

(Processor Energy Model Calculation using Heat Equilibrium)

전 상 우 † 김 현 희 † 김 지 홍 ††
(SangWoo Jun) (HyunHee Kim) (Jihong Kim)

요 약 최근 마이크로프로세서의 공정 기술이 발전함에 따라 프로세서 칩의 전력 밀도가 높아지고, 그 결과로 발열량이 증가하여 신뢰성을 떨어뜨리며, 성능에도 영향을 끼치고 있다. 기존 연구에서는 하드웨어 성능 카운터를 사용한 전력과 발열 모델을 사용하여 예측하고 대처하는 방법으로 이 문제를 해결하고자 하였다. 하지만 초기에 모델을 정립할 때는 별도의 기기를 통해 수동으로 전력을 측정해야 하므로 그 결과들이 널리 사용되기 어려웠다. 따라서 본 연구에서는 일정한 작업을 반복하는 프로세서가 도달하는 열 평형 상태의 특성을 바탕으로 별도의 기기 없이 전력과 발열 모델을 자동으로 계산하는 방법을 고안하였다. 이렇게 계산한 발열 모델로 온도를 예측한 결과 평균 1°C 내외의 오차가 관찰되었다. 모델의 계산이 용이해지면서 이를 이용한 방안들이 널리 사용될 수 있을 것으로 예상된다.

키워드 : 열평형, 전력모델, 발열모델

Abstract As the manufacturing technology of microprocessors advance, the power density of chips have also increased. This has led to an increase of heat dissipation which is having negative affects on reliability and performance. Previous research has attempted to solve this problem by forecasting and managing heat dissipation via an energy consumption model based on Hardware Performance Counters. However, such result could not be widely applied because the initial calculation of the model for individual systems required additional external machinery and human work. This paper describes a novel method of calculating the power consumption model without any additional hardware through regression-analyzing various processor states when its temperature is steady. When used to predict core temperatures, results from interim models showed less than 3 degress of error. By making the model calculation easy and automatic, it will be possible to widely use solution utilizing such methods.

Key words : Power Model, Thermal Model, Multicore

1. 서 론

최근 마이크로프로세서의 공정 기술이 발전함에 따라 프로세서 칩의 전력 밀도가 프로세서 성능을 발전시키는데 주요 제약사항이 되고 있다. 특히 높은 전력 밀도로 인해 발열량이 증가하게 되는데, 이는 칩의 신뢰성을 떨어트리고 성능에도 영향을 줄 뿐만 아니라 온도를 낮추기 위한 쿨링과 패키징 비용도 점점 증가시킨다.

여러 기존 연구[1,2]에서 하드웨어 성능 카운터를 사용한 전력 모델을 만들어서 각 태스크의 전력 소모와 발열 양상을 예측하고, 이에 대응할 수 있도록 프로세서의 속도를 제한하거나(throttling) 스케줄링하는 기법 등을 제안하였다. 그리고 초기에 전력 모델을 정립하기 위해 여러 특성을 갖는 벤치마크를 실행하면서 실제 전력 소모를 측정하고, 이렇게 얻은 정보를 회귀분석하는 방

· 본 논문은 정부(교육과학기술부)의 재원으로 한국과학재단과 한국연구재단-미래기반기술개발사업(첨단융복합분야)의 지원을 받아(No. 2010-0020724) 수행되었습니다.
· 이 논문은 제37회 추계학술발표회에서 '열 평형 상태를 이용한 프로세서 전력 모델 계산'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 서울대학교 컴퓨터공학과
aradia4@snu.ac.kr
hh0726@davinci.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학과 교수
jihong@davinci.snu.ac.kr

논문접수 : 2010년 12월 15일
심사완료 : 2011년 2월 1일

식이 주로 사용되었다. 이렇게 계산된 모델은 실제 전력 소비나 온도 변화를 예측하는데 쓰이기도 하고, 전력 소모나 발열을 고려한 스케줄링 등의 기법의 효과를 확인하거나 어떤 프로그램의 전력 소비 양상을 정적 분석하는데 사용되기도 한다[3].

다만 이러한 모델의 인자들은 특정 시스템이 사용하는 팬의 성능이나 주위 온도 등에도 영향을 받기 때문에 정확한 모델을 얻기 위해서는 개별적인 시스템에 대해 계산을 행해야 한다. 하지만 기존의 모델 계산 방법에는 방법은 멀티미터 등의 외부 기기를 사용하여 수동으로 전력을 측정하는 작업이 필요하기 때문에 널리 사용될 수 없다는 제한이 있었다.

본 논문에서는 열 평형 상태에 도달한 물체에는 들어가는 열량과 빠져 나오는 열량이 동일하다는 사실에 착안하여, 일정한 성질의 작업을 반복하는 벤치마크를 실행하면서 프로세서가 도달하는 평형 상태를 측정하여 전력과 발열 모델을 계산하는 방법을 제안하고자 한다. 이 방법은 프로세서에 주어진 자원들만을 사용하여 모델을 계산할 수 있으며, 모든 과정을 완전히 자동화할 수 있다는 장점이 있다. 실험 결과 본 논문에서 제안하는 방법으로 계산한 전력 모델이 실제 벤치마크 상황에서 평균 1°C 내외의 오차 수준에서 적용되는 것을 확인할 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구의 배경이 되는 정보를 살펴보고, 3장에서는 실제로 리눅스 시스템에서 전력과 발열 모델을 계산하는 과정을 설명한다. 그리고 4장에서는 계산된 모델의 성능을 분석한다. 마지막으로 5장에서는 본 논문의 결론과 추후 연구의 방향을 논의한다.

2. 배 경

2.1 성능 카운터 기반의 전력 모델

시스템 전반이나 태스크의 전력 소모 및 발열량과 관련된 특성을 분석하기 위해서는 시스템의 전력 소모 및 발열량을 평가하기 위한 예측 모델이 존재해야한다. 여러 선행 연구[1,2]에서는 전력 소모 예측을 위해 최근 대부분의 플랫폼에서 제공하는 하드웨어 성능 카운터의 값들에 기반한 전력 모델을 제안하였다. 즉, 각 성능 카운터 값에 프로세서가 소비하는 전력이 일정하다고 가정하고, 식 (1)과 같은 모델을 만드는 것이다. 모델의 정립을 위해서는 태스크의 수행 중에 하드웨어 성능 카운터의 값들(예를 들어 수행된 명령어 개수, L1 캐시 접근 수, L2 캐시 접근 수, 데이터 지연 사이클 수 등)을 측정하고, 이 값들과 실제 측정된 전력 소모를 기반으로 선형 회귀분석을 행한다. 그리고 전력 모델 정립 과정에서 각 계수가 결정되고 나면, 앞으로는 성능 카운터를 통해 프로세서가 행한 작업을 알 수 있다면 그 과정에서 사용한 전력량도 예측할 수 있게 된다.

$$Energy = P \times Count_{Instruction} + Q \times Count_{CacheMiss} + \ldots$$

식 (1) 성능 카운터 기반의 전력 모델

위에서 $Count_{Instruction}$ 은 프로세서가 처리한 명령의 갯수를 뜻하고, $Count_{CacheMiss}$ 는 성능 카운터가 측정한 캐시 미스의 횟수를 뜻한다. 즉, 프로세서가 소비하는 에너지는 프로세서가 처리하는 명령의 갯수와 캐시 미스의 횟수 등의 이벤트의 횟수에 비례하는 것이다. 본 연구에서도 식 (1)과 같은 하드웨어 성능 카운터 기반의 전력 모델을 사용한다.

2.2 열 평형 상태의 특징

일정한 작업을 행하고 있는 코어의 온도가 일정하게 유지되고 있다면, 해당 코어는 열 평형 상태에 도달한 것이다. 이 상태에는 코어가 사용한 전력에 의한 발열과, 주위 코어들로부터 유입되는 열, 그리고 팬을 통해 확산되는 열량의 합이 0이 되게 된다. 예를 들어 쿼드 코어 프로세서의 0번 코어에 대해서 식 (2)와 같은 공식이 적용되게 된다. 본 논문에서 코어는 온 칩 캐시를 포함하는 의미로 사용된다.

$$\begin{aligned} & A \times Temperature_{Core0} + B \times Speed_{Fan} \\ & = P \times Count_{Instructions} + Q \times Count_{CacheMiss} + \ldots \\ & + X \times TempDiff_{Core1} + Y \times TempDiff_{Core2} \\ & + Z \times TempDiff_{Core3} + Const \end{aligned}$$

식 (2) 열 평형 상태의 열 입출력

위 식에서 좌변은 코어에서 발산되는 열량을 뜻하며, 우변은 코어로 유입되는 열량을 뜻한다. 코어에서 발산되는 열량은 코어의 온도($Temperature_{Core0}$)와 팬의 회전 속도($Speed_{Fan}$)에 비례하며, 코어로 유입되는 열량은 식 (1)에서 알 수 있는 전력 소모량과, 다른 코어들과의 온도차($TempDiff_{Core[n]}$)에 비례하는 것이다.

여기서 P, Q와 같은 인자들의 값은 각 작업당 에너지 소모량인 식 (1)에서의 P, Q 인자들과 같은 의미를 가지므로, 이 식을 풀면 전력 소모 모델을 얻을 수 있게 된다.

2.3 팬의 속도에 따른 열 확산

식 (2)는 팬의 속도와 팬에 의한 열 확산이 선형 관계에 있다는 사실을 전제로 한다. 다른 상황들을 고정시키고 팬 속도만을 조정하면서 실험해 본 결과, 그림 1과 같이 팬의 회전 속도와 평형 온도는 선형 관계에 있으며, 따라서 제안된 모델의 가정이 성립함을 확인할 수 있었다.

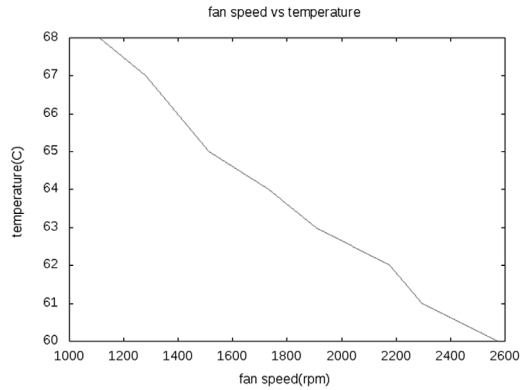


그림 1 팬 속도에 따른 평형 온도

3. 전력 모델의 계산

본 연구에서는 모델의 단순함을 유지하기 위해 식 (3)와 같이 각 코어가 처리한 명령의 갯수와, 각 코어에서 일어난 캐시 미스의 갯수를 사용한 전력 모델을 사용하였다. 이 두 이벤트를 선택한 이유는 코어의 전력 소모에 이 둘이 가장 큰 영향을 끼치는 것으로 알려졌기 때문이다. 차후 연구에서는 부동 소수점 연산의 갯수 등 더 많은 인자들을 사용하여 더 정확한 모델을 계산할 것이다.

$$Temperature_{Stable} + B \times Speed_{Fan} = P \times Count_{Instructions} + Q \times Count_{CacheMiss} + X \times TempDiff_{Core1} + Y \times TempDiff_{Core2} + Z \times TempDiff_{Core3} + Const$$

식 (3) 열 평형 상태의 모델

식 (2)의 식에서 A를 양변에 나누어 변수를 하나 제거할 수 있었으며, 따라서 7가지 열 평형 상황의 정보를 수집하면 필요한 7개 인자들의 값을 계산할 수 있다. 이를 위해 짧은 작업을 계속 반복하는 hit_cache(H)와 miss_cache(M)의 두 가지 벤치마크를 만들고, 표 1과 같은 조합으로 실행하여 데이터를 수집하였다. Set 0, 1, 2는 코어에서 실행하는 작업의 성질을 다르게 하여 전력 소모와 관련된 인자들을 파악할 수 있도록 하며, Set 3, 4, 5는 다른 코어들의 온도를 다르게 하여 코어들 사이의 열 확산과 관련된 인자들을 파악할 수 있게 하고, Set 6은 팬의 속도를 다르게 하여 팬에 의한 열 확산 인자를 파악할 수 있게 한다.

본 연구에서는 데이터 수집을 위해 리눅스의 Core-Temp 모듈을 사용하여 2초 간격으로 프로세서의 온도 측정 다이오드를 읽어서 각 코어의 온도를 기록하였다.

표 1 모델 계산을 위한 테스트셋의 구성

Set	Core 0	Core 1	Core 2	Core 3	FanSpeed
0					50%
1	M				50%
2	H				50%
3	H	H			50%
4	H		H		50%
5	H			H	50%
6					100%

M(miss_cache): 캐시 미스가 많은 벤치마크
H(hit_cache): 캐시 미스가 적은 벤치마크
빈칸: 아무것도 실행하지 않음

프로세서가 제공한 온도 측정 다이오드는 1°C 단위의 정보만을 제공하기 때문에, 40초 이상 1°C 간격의 온도를 유지한다면 열평형으로 간주하는 방식을 사용하였다.

이렇게 구한 데이터는 바로 계산하여 인자들을 파악할 수도 있으며, 더 신뢰할 만한 식을 얻고자 한다면 여러 차례 데이터를 수집하여 회귀 분석을 행할 수도 있다. 본 연구에서는 위 테스트셋을 4회씩 반복한 다음 회귀분석하는 방법을 사용하였다.

이 과정을 통해 Intel Quad-Core에서 계산한 모델의 예는 표 2에서 확인할 수 있다. 표에서 Core[n] 항목들은 코어 사이의 열 확산 인자들(식 (3)에서 X, Y, Z)이며, Instruction과 CacheMiss는 각각 P와 Q, Fan은 팬 속도에 대한 인자(B), Constant는 상수(C)를 뜻한다. 여기서 구한 인자들을 식 (1)에 대입하면 코어의 전력 모델을 얻을 수 있으며, 식 (3)에 대입하면 코어의 발열 모델을 얻을 수 있다.

표 2 Intel Quad-Core의 모델 인자

	Core0	Core1	Core2	Core3
Core0	0	-	-	-
Core1	1.56	0	-	-
Core2	1.68	0.27	0	-
Core3	1.60	1.80	0.75	0
Instruction	1.1×10^{-5}	1.2×10^{-7}	3.3×10^{-7}	3.3×10^{-7}
CacheMiss	8.8×10^{-4}	2.4×10^{-5}	1.0×10^{-4}	3.0×10^{-5}
Fan	6.8	4.1	4.0	4.7
Constant	65840.4	59584.3	57711.3	53840.0

앞에서 계산한 인자들을 사용하여 임의의 2초 구간에 대해 식 (3)을 실제로 계산하면 코어의 열 상태를 파악할 수 있다. 좌우변의 값이 일치하지 않을 경우는 열 평형 상태가 아닌 것이며, 좌변에 남는 값이 코어에 잔류하고 있는, 즉 코어의 온도를 올리게 될 열량을 의미한다. 매 구간 프로세서의 온도는 이 잔류 열량에 비례하여 변화하며 평형 상태로 다가가게 된다.

4. 모델의 성능 분석

모델의 계산과 성능 분석에 사용된 시스템은 Intel Core 2 Quad Q9400과 Linux 커널 2.6.32로 구성되었다. 온도 차이를 확실히 볼 수 있도록 팬 속도는 50% 수준으로 줄였으며, 모델의 단순화를 위해 Intel의 Dynamic Frequency Scaling 기법인 Speedstep을 사용하지 않도록 설정하였다.

본 논문이 제안하는 방법의 검증을 위해 실제로 계산한 모델을 통해 코어의 온도 변화를 예측한 것과 실제 측정값을 비교하는 방식을 사용하였다. 차후 연구에서 인자들을 더 사용하여 더 정확한 모델을 계산할 경우에는 모델이 계산하는 전력 소비와 실제 전력 소비 양상을 비교하는 방법을 사용할 것이다.

테스트용 프로그램으로는 Parsec Benchmark Suite [4]를 사용하였으며, 3개의 코어에서 bodytrack과 facesim 벤치마크를 각각 100초씩 실행하여 전력 사용에 의한 온도 변화뿐만 아니라 주변 코어의 온도의 영향도 확인할 수 있도록 하였다. 그림 2는 facesim에 대해 각 코어에서 0.5초마다 실제로 측정한 값(measured[n])과 모델이 예측한 값(calculated[n])을 비교하여 표현한 것이며, 표 3은 각각 facesim, bodytrack과 canneal에서의

표 3 오차와 오차 편차 정보

(a) facesim

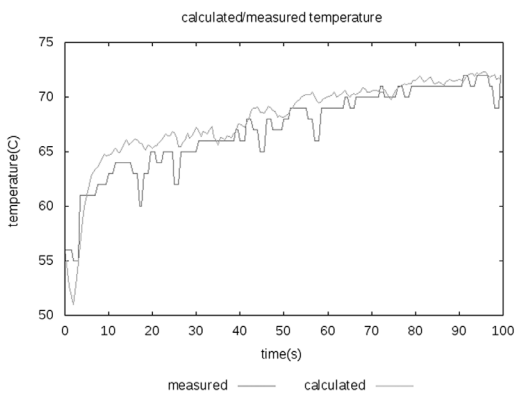
	Core0	Core1	Core2	Core3
|err|	1.24	0.90	1.31	1.14
dist	2.80	1.90	2.45	2.11

(b) bodytrack

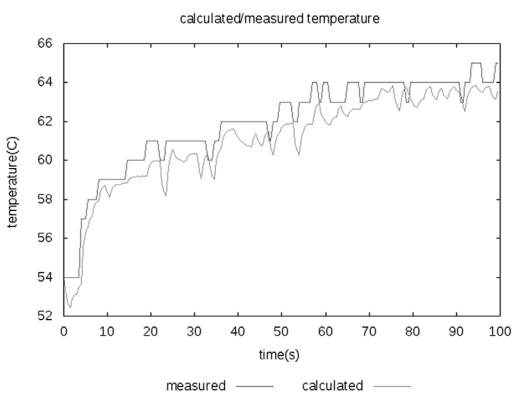
	Core0	Core1	Core2	Core3
|err|	0.87	0.57	1.37	1.84
dist	1.24	0.66	2.66	4.59

(c) canneal

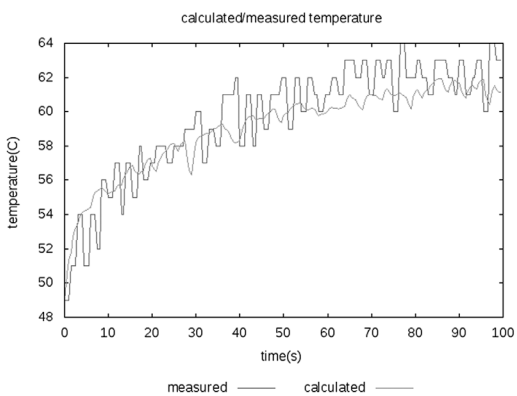
	Core0	Core1	Core2	Core3
|err|	1.80	0.63	0.77	0.86
dist	4.29	0.60	0.98	1.11



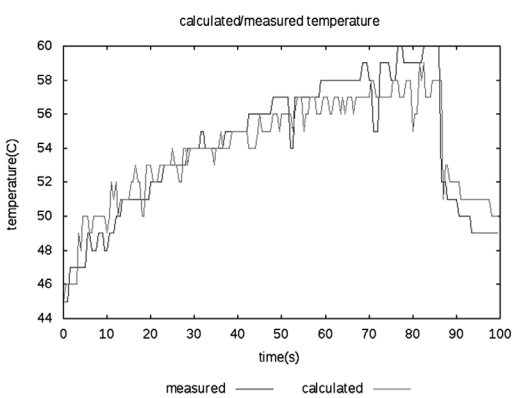
(a) Core 0의 예측값과 실측값 비교



(b) Core 1의 예측값과 실측값 비교



(c) Core 2의 예측값과 실측값 비교



(d) Core 3의 예측값과 실측값 비교

그림 2 모델 예측값과 실측값의 비교

오차와 오차의 분산 정보이다. 예측 온도의 값은 맨 처음에만 CoreTemp에서 읽은 값으로 맞추었으며, 그 뒤로는 100초동안 전혀 수정 없이 예측하였다.

100초동안 비교한 결과 각 코어의 오차 수치는 표 3에서 확인할 수 있다. 표에서 |err|는 오차 절대값의 평균을 뜻하며, dist는 오차의 분산을 뜻한다. 오차가 2°C 안을 유지함을 확인할 수 있다.

5. 결 론

기존의 연구들은 정학한 전력 모델을 측정하여 이를 적용할 수 있는 방안들에 집중하였다. 하지만 발열 모델의 경우 해당 시스템이 사용하는 팬의 특성이나 주위 기온 등의 요인에 의해 시스템마다 다른 값들을 가질 수 있으며, 전력 모델의 경우도 주변 온도나 전원 특성의 변화 등의 이유로 인자가 변화하는 경우가 있을 수 있다. 따라서 기존 연구들에서 제시한 수동적인 계산 방법은 널리 사용되기에는 너무 번거로운 단점이 있었다.

본 논문에서 제시한 전력 및 발열 모델 계산 방식은 일반적인 시스템에서 제공하는 자원들만을 사용하여 완전히 자동화된 방법으로 필요한 정보를 수집할 수 있다는 장점이 있으며, 실험 결과 만족할 만한 온도 예측 능력을 보여주었다. 따라서 수동으로 모델을 계산할 전문성을 갖지 못한 일반 사용자나 많은 수의 시스템들이 배치되어야 하는 데이터센터에 발열 모델을 사용한 기법들이 적용되기 쉽게 해줄 것이다. 또한 자주 환경이 바뀌는 로봇이나 모바일 디바이스 등에서도 전력과 발열 관리에 유용하게 사용될 것으로 예상된다.

앞으로 본 논문의 모델 계산 방식을 확장하여, 이미 사용 중인 시스템에서 측정값과 예측값의 차이를 이용하여 모델의 오차를 줄이는 방식을 고안할 수 있다면 모바일 디바이스에서 전력 및 발열 관리에 유용할 것으로 생각된다. 더불어, 이렇게 계산된 모델을 사용하여 전력 사용량과 발열 수준을 정확하게 관리할 수 있는 스케줄러의 연구도 진행할 계획이다.

참 고 문 헌

[1] Bellosa, F, "The benefits of event: driven energy accounting in power-sensitive systems," In *Proceedings of the 9th Workshop on ACM SIGOPS European Workshop: Beyond the Pc: New Challenges For the Operating System*, pp.37-42, 2000.

[2] Choi, W., Kim, H., Song, W., Song, J., and Kim, J., "ePRO-MP: A tool for profiling and optimizing energy and performance of mobile multiprocessor applications," *Scientific Programming*, vol.17, Issue 4, pp.285-294, 2009.

[3] Kevin Skadron, Mircea R. Stan, Karthik Sankaranarayanan, Wei Huang, Sivakumar Velusamy, and David Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Archit. Code Optim.* 1, 1 (March 2004), pp. 94-125, 2004.

[4] Bienia, C., Kumar, S., Singh, J. P., and Li, K., "The PARSEC benchmark suite: characterization and architectural implications," In *Proceedings of the 17th international Conference on Parallel Architectures and Compilation*, pp.72-81, 2008.

전 상 우

2011년 서울대학교 컴퓨터공학부 학사. 관심분야는 칩 멀티 프로세서 아키텍쳐, 병렬 프로그래밍 모델

김 현 희

2004년 중앙대학교 컴퓨터공학부 학사 2006년 서울대학교 전기컴퓨터 공학부 석사. 2006년~현재 서울대학교 전기, 컴퓨터 공학부 박사과정. 관심분야는 칩 멀티 프로세서 아키텍쳐, 저전력 시스템, 임베디드 소프트웨어

김 지 홍

1986년 서울대학교 계산통계학과 학사 1988년 University of Washington 컴퓨터과학과 석사. 1995년 University of Washington 컴퓨터과학 및 공학과 박사 1995년~1997년 미국 Texas Instruments 선임연구원. 1997년~현재 서울대학교 전기·컴퓨터공학부 교수. 관심분야는 임베디드 소프트웨어, 저전력 시스템, 멀티미디어 시스템, 컴퓨터 구조