멀티 코어 프로세서의 온도관리를 위한 방안 연구 및 열-인식 태스크 스케쥴링

(Thermal Management for Multi-core Processor and Prototyping Thermal-aware Task Scheduler)

최 정 환 †

(Jeonghwan Choi)

요 약 최신의 마이크로프로세서 설계에서는 전력 관련 문제들이 중요한 고려사항이 되었다. 온-칩(On-chip) 온도 상승은 이와 관련하여 중요한 요소로 부각 되었다. 이를 적절하게 처리하지 않을 경우 냉각 비용과 칩 신뢰성에 부정적인 결과를 초래한다. 이 논문에서 우리는 시간적/공간적인 핫 스폿(Hot spot) 완화를 위한 설계들과 열 시간 상수, 작업부하 변동, 마이크로프로세서의 전력 분배 사이의 보편적인 상충관계(Trade off)들을 조사한다. 우리의 방안은 작업부하의 실행위치/순서를 변경하고 동시실행 스레드의 수를 조절하여 시스템의 공간 및 시간적인 열 틈새(Heat slack)에 영향을 줌으로써, 운영체계(OS)와 이미 시스템에 존재하는 하드웨어의 지원만으로 적절한 시간제한내에 작업부하를 조절함으로써 온-칩 온도를 낮출 수 있다.

키워드 : 전력관리, 열관리, 스케쥴링

Abstract Power-related issues have become important considerations in current generation microprocessor design. One of these issues is that of elevated on-chip temperatures. This has an adverse effect on cooling cost and, if not addressed suitably, on chip reliability. In this paper we investigate the general trade-offs between temporal and spatial hot spot mitigation schemes and thermal time constants, workload variations and microprocessor power distributions. By leveraging spatial and temporal heat slacks, our schemes enable lowering of on-chip unit temperatures by changing the workload in a timely manner with Operating System (OS) and existing hardware support.

Key words : System Level Power Management, Thermal Management

1. 서 론

지속적인 마이크로프로세서 설계기술의 진보로 시스템의 집적도와 성능이 높아졌다. 하지만 이에 따른 부작용으로 현재세대 프로세서는 심각한 전력밀도 문제와 이로 인한 온도("Hot spot") 문제를 유발한다. 높아진 최고온도와 평균온도로 인하여 칩과 시스템의 수명이 짧아졌다. 연구에 의하면 작동온도가 10-15도씨 상승하면, 전기회로의 수명은 반으로 줄어드는 것으로 나타났다[1]. 또한, 총 전력과 프로세서상의 최고온도의 상승은 열 분산을 위한 냉각비용과 패키징비용을 증가시킨다[2]. 더구나 전력과 전력밀도 값이 더 높아 질수록 비용 증가의 폭은 더 커진다. 전력과 온도에 관련한 제약들로 인하여 코어주파수 증가는 크게 제한되었고, 이에 따라 낮은 작동주파수로 고성능을 제공할 수 있는 새로운 멀티-코어 칩 설계방식이 도입되었다. 그러나 프로세서 생산회사들이 좋은 단일프로세서 성능을 유지하면서도 칩 전체의 성능을 증가시키려는 노력을 하는데 있어, 각 코어 부분 내의 국소적 핫 스폿(Hot spot)이 여전히 문제가 되고 있다.

· 본 연구는 국토해양부 첨단도시기술개발사업 - 지능형국토정보기술혁신사업과제의 연구비지원(07국토정보C03)에 의해 수행되었습니다.

† 정 회 원 : Penn State University Computer Sci & Eng Post-Doc
jhchoi@gmail.com

논문접수 : 2008년 3월 11일
심사완료 : 2008년 4월 28일

과거에는 최고온도를 제한하기 위한 하드웨어 솔루션으로 동적 열관리(Dynamic Thermal Management) 기술들을 사용했다[3]. 이 기술들은 온도가 사전에 설정한 임계 값에 도달하면, 프로세서의 동작을 멈추게 하여, 전력소비를 낮춘다. 전역 클럭 게이팅(Global clock gating), 클럭 쓰로틀링(Clock-throttling), 전압/주파수 스케일링 등의 칩의 온도가 제한온도에 이르는 것을 효과적으로 방지하기 위한 다양한 방안들이 제안되었다. 그러나 이러한 급작스러운 하드웨어 차단방법들은 매우 높은 성능이 필요한 일부 애플리케이션들의 성능을 심하게 떨어뜨릴 수 있다. [2]에 설명된 대로, 멀티-코어 칩 전체에 걸친 보다 더 균등한 전력소비 배분을 강화하는 접근법이, 성능을 떨어뜨리지 않으면서도 열 분산 문제를 완화시키는 해결책으로 선호되고 있다.

그러나 작업부하 변화에 효율적으로 대처하여 성능의 손실을 최소화하고 전력과 열 제한에 대응할 수 있으려면 개선된 동적 전력재분배 기술들이 필요하다.

이 논문에서 우리는 실제 1.2 GHz POWER5 시스템에서 실시된 실험을 배경으로(시간적 및 공간적) 핫 스폿 완화설계에 있어서의 열 시간 상수, 작업부하 변화, 칩-수준 전력 배분 사이의 상충 관계를 조사한다.

두 가지 관찰이 우리의 연구를 가능하도록 하였다: (a) 현대 프로세서에 널리 사용되는 정밀 클럭 게이팅(Fine-grain Clock-gating) 기술 관점에서 볼 때, 칩 상의 온도분포 프로파일과 전력 프로파일은 칩 내의 단위요소의 이용 상태와 긴밀하게 연관된다. (b) 칩에서의 온도의 상승 시간과 하락시간은 대개 수백 밀리 세컨드 범위에 있다. 우리는 온-칩 핫 스폿 문제를 완화시키기 위해 운영체계수준에서의 온도-인식 스케쥴링의 효과를 조사했다. 이 개념을 이용하여 리눅스 시스템에서 열-인식 스케쥴러에 구현했다. POWER 5 프로세서 기반의 리눅스 시스템에서 실제 온도측정과 실행 시간측정으로 우리는 이 같은 접근법이 가지는 두 가지 이점을 증명할 수 있다: (a) 추가적인 하드웨어 비용이 발생하지 않고 (b) 사용자 입장에서 성능에 큰 손실 없이 동적 온-칩 전력 재분배를 성취하며 이와 동시에 마이크로-아키텍처 시스템 수준에서 결정된 정보를 이용할 수 있다.

이 논문의 나머지 부분은 다음과 같이 구성된다. 제2장에서는 운영체계수준의 열 완화를 위한 접근법의 타당성 조사 결과를 제시한다. 제3장에서는 우리의 리눅스 기반의 열-인식 스케쥴러의 구현을 실험 결과와 함께 설명한다. 제4장에서는 관련연구들을 설명한다. 마지막으로 제5장에서는 향후 연구에 관한 간략한 설명과 함께 결론을 제시한다.

2. POWER5 열 완화 연구

우리는 원형 리눅스(Bare Metal Linux)를 운영하는 1.2GHz POWER 5 시스템에서 실험을 실시했다[4]. 온-칩 온도 변화를 측정하려고, 원형 리눅스를 변형하여 매 스케쥴러 눈금(tick)마다 4밀리 세컨드의 정밀도로 24개의 온 칩 열 센서를[4] 이용하여 칩 온도를 측정하였다. Hamann등이 개발한 Spatially-resolved Imaging of Microprocessor Power (SIMP) 방법을 이용하여 열 센서를 미세조정하였다[5]. SIMP 방법은 POWER5 칩이 안정적인 온도에 도달할 때 칩의 적외선 이미지를 포착하여 이미지로부터 측정된 온도들과 디지털 온도 센서 계측 값들을 비교함으로써 센서를 조정한다[4]. 적외선 이미지 포착이 가능하도록, 우리는 POWER 5 시스템의 금속 방열판을 투명한 액체 발열판으로 대체했다. 표 1에는 측정된 POWER 5의 구성이 나와 있다.

표 1 POWER 5 구성

Processors	Two 1.2GHz out-of-order core	L1 Dcache	32KB, 4way
H/W threads	Two threads per core	L1 Icache	64KB, 2way
Memory Size	2GB	L2	1.44MB, no L3

2.1 열 시간 상수 연구

POWER 5 칩에서의 열 시간 상수를 조사하고자, 우리는 코어 1에 고정한 고온의 마이크로-벤치마크(daxpy)의 시작 시각과 종료 시각을 기록했다. 그림 1에서, 왼쪽에 있는 그래프는 시간이 경과하면서 각 열 센서에 의해 측정된 리눅스 유휴 루프(Idle loop)로부터의 상대 온도의 변화를 보여준다. 가운데에 있는 그래프와 오른쪽에 있는 그래프는 동일한 벤치마크에 대해서 온도가 올라간 때와 떨어진 때를 확대시킨 것이다. 왼쪽 그림에서 우리는 작업부하의 변화에 의해 온-칩 온도가 16도 씨 까지 변할 수 있다는 것을 관측했다. 가운데 그림과 왼쪽 그림 두 그림들로부터 우리는 온도의 상승 시간과 하락 시간은 수백 밀리 세컨드의 범위인 것을 관측 했다. 이러한 시간 상수들은 전형적인 운영체계 스케쥴러 눈금(Tick)인 10ms 보다 훨씬 더 크기 때문에, 소프트웨어(예, OS 또는 Hypervisor)에서 온-칩 열 관리를 실행할 수 있으며 시스템이 더 높고 위험한 열 상태에 도달하기 전에 반응할 수 있음을 시사한다.

2.2 코어간 반복이동(Core hopping) - 공간적 열틈새(Heat slack)의 활용 가능성

운영체계 스케쥴러가 공간적인 열틈새를 활용함으로써 칩 내의 핫 스폿을 완화시킬 수 있다는 것을 보이려고 우리는 단일 스레드 프로그램을 듀얼-코어 POWER 5 상에서 4밀리 세컨드마다 두 코어에서 번갈아 가며 이동하여 실행하도록 바꾸었다. 한 코어는 프로그램을 가동하고 이 때 다른 코어는 리눅스 유휴 루프(Linux idle loop) 또는 다른 데몬(daemon) 태스크들을 실행한

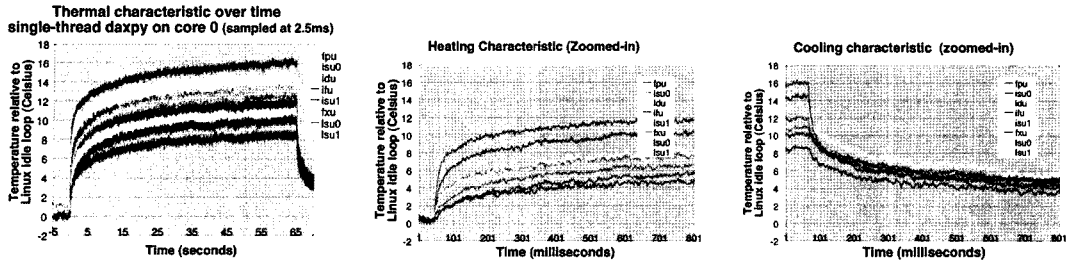


그림 1 프로세서의 온도변화는 운영체계 시간 구획들 보다 더 오랜 시간에 걸쳐 읽어난다.

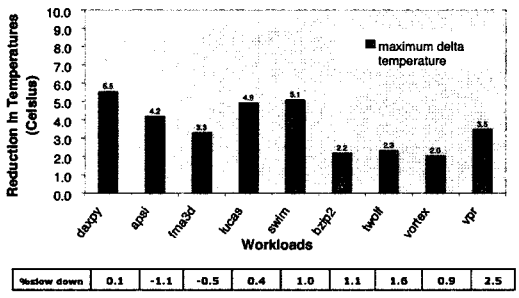


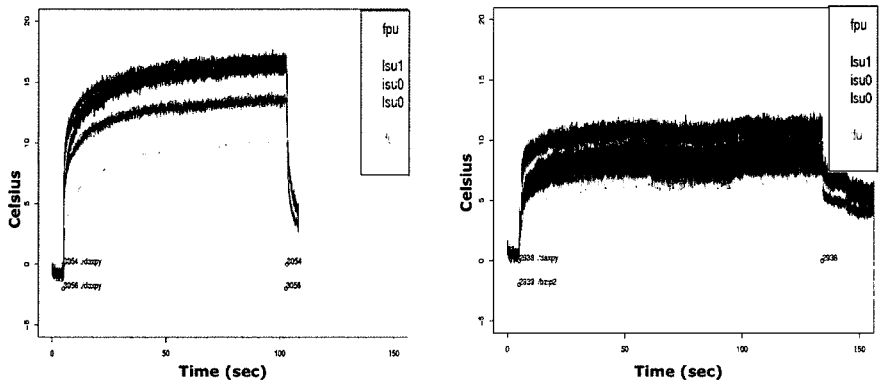
그림 2 코어간 반복이동은 작은 양의 성능감소를 일으키며 온-칩 온도를 줄인다.

다. 그림 2는 핫 스폿의 감소정도와 반복이동이 실행시간에 미치는 영향을 보여준다. 이 그림으로부터, 우리는 코어간 반복이동이 유발하는 평균 성능감소는 1% 미만이고, 5.5도씨의 최고온도 변화를 일으킬 수 있는 것을 관측했다. 그림 2에 나온 프로그램들 가운데 두 개에서는 실제로 속도가 약간 증가한 것을 보았다; 이것 들은 아마도 가벼운 실험 잡음을 반영하는 것일 수도 있다. 이러한 잡음을 제외시키면 1.08% 평균 감속이 관측되었고, 성능에 대한 영향은 적다. 두 코어들이 공통된 온-칩 L2 캐시를 공유하므로, 코어간 반복이동은 비용이 비싼 추가적인 L2 캐시 손실을 일으키지 않고, 4ms의 시간은 충분히 커서, L1 캐시 워밍업에 들어간 성능 비용은 미세한 영향을 준다. 그 결과 성능 저하는 3% 미만으로 측정되었다.

2.3 태스크 스케줄링 - 시간적 열틈새의 활용 가능성

운영체계 스케줄러가 지능적인 작업부하 스케줄링을 하여 시간적 열틈새를 잘 활용함으로써 핫 스폿을 완화시킬 수 있다는 것을 보이기 위해서 우리는 POWER5 상에서 서로 다른 열 특성들을 나타내는 SPEC2K 프로그램들을 혼합하고 또한 리눅스 명령 taskset을 사용하여 모든 태스크들이 동일한 코어의 동일한 하드웨어 스레드에 고정되도록 했다.

그림 3은 동일한 하드웨어 스레드를 공유하는 두 벤치마크를 가동할 때에 시간 경과에 따른 온도 변화를 보여준다. 그림 3의 왼쪽 그래프는 두 개의 고온형 태스크들(daxpy)의 가동을, 오른쪽 그래프는 한 개의 고온의 태스크(daxpy)와 한 개의 저온형 태스크(bzip2)의 가동을 보여준다. 그림 3으로부터 우리는 한 개의 저온형 태스크와 한 개의 고온형 태스크를 혼합하는 것이 온-칩 온도를 5도씨나 내리는 것을 관측했다. 4 밀리



(a) FP/FP (*daxpy/daxpy*) (b) FP/Integer (*daxpy/bzip2*)

그림 3 서로 다른 열특성을 가지는 태스크를 같이 실행시킴으로서 온-칩 핫 스폿을 감소시킨다.

세컨드 간격으로 운영체계의 간섭을 필요로 하는 코어 간 반복이동과는 달리, 싱글스레드 모드에서 고온형 태스크들과 저온형 태스크들을 잘 혼합시키면 운영체계의 중재 없이 오랜 시간 동안 온도를 낮출 수 있다.

2.4 동시 멀티스레드(SMT) 스케쥴링 - 시간적 열틈새의 활용 가능성

운영체계 스케쥴러가 동시 멀티스레드-모드 태스크를 지능적으로 공동-스케쥴링 함으로써 온도를 줄이는 효과가 있다는 것을 보이려고, 우리는 리눅스 명령 task-set을 이용하여 각 태스크를 동일한 코어의 서로 다른 하드웨어 스레드에 고정하였다. 그림 5는 동일한 코어를 공유하지만 동시 멀티스레드에서 각자 독립된 하드웨어 스레드를 사용하는 두 벤치마크를 가동했을 때 시간 경과에 따른 상대 온도 변화를 보여준다. 그림 4의 왼쪽 그래프는 두 개의 고온 태스크들(lucas와 swim)의 가동을 보여주고 오른쪽 그래프는 한 개의 고온 태스크(lucas)와 한 개의 저온 태스크(vortex)의 가동을 보여준다. 그림 4로부터, 우리는 동시 멀티스레드에서 한 개의 저온 태스크와 한 개의 고온 태스크를 혼합하는 것이 온-칩 온도를 3도씨 낮추는 것을 관측하였다.

3. 리눅스에서 열-인식 스케쥴러 프로토타입 개발(PROTOTYPING)

2장에서, 실제 POWER5 시스템에서 실시한 우리의 실험들은 스케쥴러 눈금의 단위에서 방안들을 실행하여 시스템 하이퍼바이져 또는 운영체계에서의 칩-수준 핫스폿 완화 기술들의 실행가능성을 보여주었다. 리눅스와 같은 현대 운영체계는 추가적인 복잡성 없이 이러한 핫스폿 완화 기술을 실행할 수 있기 때문에, 우리는 먼저 프로그램의 가능성을 평가한 후, 열-인식 스케쥴러의 리눅스 커널(Linux-2.6.17)기반 프로토타입을 개발했다. 우리는 이 연구에서 지연실행(Deferred execution), 냉각 루프를 활용한 스레드 감소라는 두 가지 방안을 구현 했다. 이 두 가지 기술은 트리거 조건과 반응 시간이 다르다. 이 두 가지 기술들이 모두 온-칩 온도 상승을 성공적으로 중단시키지 못할 경우, 우리는 시스템이 인출 쓰로틀링(Fetch-throttling)이나 주파수/전압 스케일링 같은 하드웨어 기반(Hardware triggered)의 온도 관리를 이용할 것이라고 가정한다.

3.1 고온 태스크 지연실행(Deferred Execution of Hot Tasks)

우리는 리눅스 스케쥴러에서 고온 태스크 지연 실행(Deferred Execution) 이라고 하는 반응적 설계를 고안했다. 코어가 다수의 태스크들을 가지고 있으며 태스크들 가운데 하나가 지속적으로 코어를 뜨겁게 할 때, 그 스케쥴러는 현재 가동되고 있는 고온 태스크의 시간 구획(Time slice)을 일시적으로 정지시켜서 고온 태스크가 코어를 더 뜨겁게 하기 전에 상대적으로 저온의 다른 태스크가 가동될 수 있도록 한다.

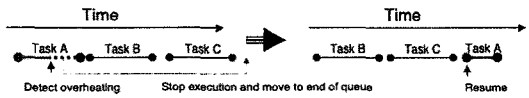
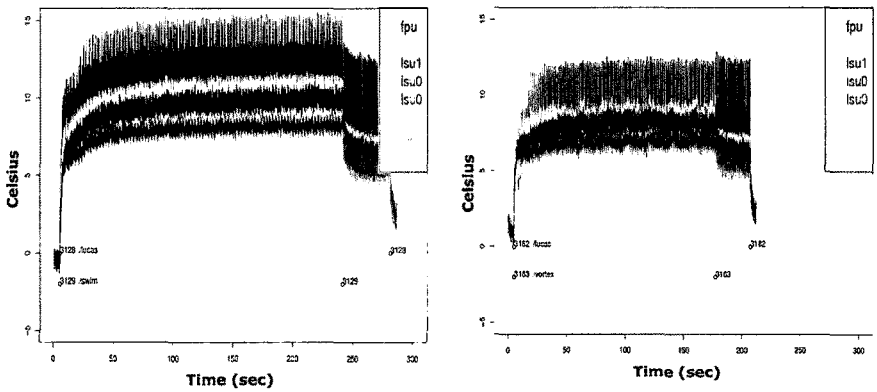


그림 5 고온 태스크 지연 실행의 예

그림 5는 고온 태스크 지연 실행의 예를 보여준다. 그림 6의 좌측에 있는 그래프에서, 시스템은 기본 리눅스 스케쥴러를 이용한다. daxpy는 빨리 가열되는 스레드이므로, daxpy가 스케쥴되는 스케쥴링 눈금에 도달할 때마다 코어의 온도를 상승 시킨다. 명확한 설명을 위해서 우리는 이 특정 작업부하에 대해 프로세서에서 가장 뜨거운 위치 인 FPU 유니트의 온도만 제시한다.



(a) FP/FP pair (*lucas-swim*) (b) FP/Integer pair (*lucas-votex*)

그림 4 동시 멀티태스크 환경에서 태스크-조합에 따라 온-칩 핫 스폿이 줄어든다.

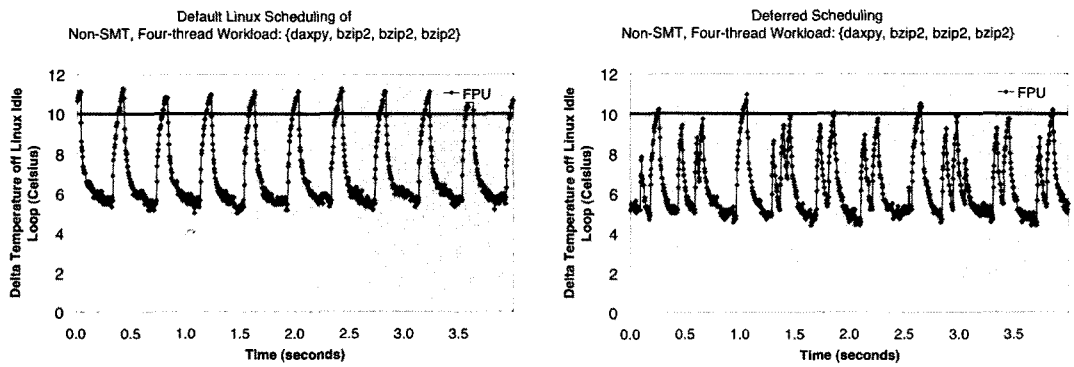


그림 6 열-인식 리눅스 스케줄러가 코어의 최대온도를 낮춘다.

그림 6의 우측에 있는 그래프에서, 시스템은 고온 태스크 지연실행을 채택한 우리의 열-인식 리눅스 스케줄러가 운영된다.

이 설계는 시스템의 온도가 임계온도에 도달하면 daxpy를 운영체계 실행큐(Run queue)의 끝으로 보내고 상대적로 차가운 다른 스레드들(bzip2s)을 가동하여 코어의 온도를 낮춘다. 그림 6 아래에 있는 그래프에서, 리눅스 스케줄러에서의 공평성(Fairness) 요건 때문에, 다른 스레드들이 모두 시간 구획을 끝마쳤치게 되면 스케줄러는 다시 고온 태스크를 가동해야 한다. 이로 인해 코어가 종종 뜨거워 질 때가 있다. 그러나 우리는 최고 온도가 10도씨 선을 넘는 경우를 극히 제한적인 경우에 발견했다.

3.2 동시 멀티스레드 및 싱글 스레드(Single-thread)를 위한 냉각 루프(Cool loop)

시스템이 고온 태스크로만 부하가 구성되었거나 과부하(Overload)된 경우, 열틈새가 충분하지 않아서 앞에서 제시한 지연실행이 온-칩 핫 스폿을 줄이는데 제한이 따를 수 있다. 이럴 경우, 시스템은 빠르게 반응하면서, 오버로드를 줄이는 방안을 이용하여 성능의 희생을 감수하면서 온도를 낮춘다. 현재의 POWER 5 프로세서는 하드웨어 기반 인출 쓰로틀링(Hardware-enabled fetch throttling)을 이용하여 특정한 제한 온도에 도달하면 작업부하를 줄여 온도의 상승을 제한한다. 패키지/냉각 시스템이 있는 POWER 5에서는, 이는 단지 시스템의 손상을 통제하기 위한 방법이며, 정상 작동 범위에서 일반적으로 작동하는 것은 아니다. 이와 달리 대부분의 Intel 마이크로프로세서들은 일정 형태의 동적 열 관리를 이용하고 있으며, 이러한 동적 열 관리에서는 열 문제들을 완화시키기 위한 메커니즘으로서 적은 비율의 고온 작업부하들도 일상적으로 하드웨어를 차단한다. 이런 이유로 인텔 프로세서는 비용이 더 적게 드는 냉각 솔루션을 이용할 수 있다. 하지만 Ghiasi 등은 [6]에서 이러한 접근법이 기아(Starvation)와 급작스러운 성능 부족(Shortfall)을 일으킬 수 있다고 하였다.

합리적인 성능을 유지하면서 온도를 낮추기 위해, 우리는 냉각 루프(Cool Loop)라고 하는 새로운 커널 태스크를 실행하여 시간적 열틈새 만들고자 했다. 냉각 루프는 운영체계 유휴 루프와 동일한 것으로 생각할 수 있지만, 우선순위(Priority)가 더 높다.

유휴 명령어(No-op instructions) 또는 인출 쓰로틀링이나 주파수/전압 스케일링이나 전력조절(power-gating)을 통해서 코어 온도를 낮추는 전력조절 명령어(Power-managing instructions)로 구성될 수 있다. 우리의 실험에서는, 냉각 루프를 측정을 통해서 충분한 냉각성능을 보여준 공 루프(Empty loop)를 이용했다. 우리의 구현에서는, 냉각 루프가 가동되고 있을 때에는 유용한 계산을 실행하지 않지만, 사용자 태스크보다는 우선순위가 더 높고 인터럽트 실행 루틴(Interrupt service routines)이나 스케줄러 보다는 낮은 우선 순위를 가진다. 스케줄러 눈금을 포함하여 인터럽트 루틴(Interrupt routines)은 일반적으로 수행 시간이 짧기 때문에 인터럽트 제공 루틴이 실행 되도록 허용하는 것이 온도 상승의 위험을 더 높이는 것은 아니다.

스케줄러는 매 눈금 마다 코어 온도를 체크하여 코어 온도가 미리 설정된 냉각 온도로 떨어질 때 사용자 태스크들을 재가동시킨다. 그림 7은 동시 멀티스레드에서 가동하고 있는 냉각 루프를 보여준다. 동시 멀티스레드에서는, 동시 멀티스레드에 있는 활성 하드웨어 스레드

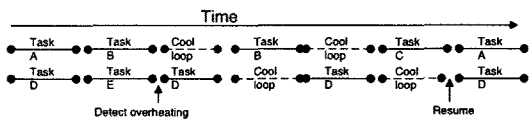


그림 7 동시 멀티스레드에서의 냉각 루프는 열을 줄이기 위해 동시 수행을 억제한다.

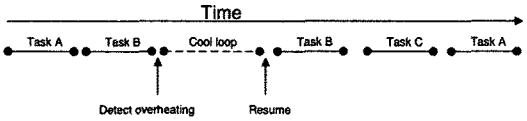


그림 8 싱글스레드를 위한 냉각 루프

의 수를 줄임으로써 온도를 낮출 수 있다. 동시 멀티스레드 코어가 뜨거워질 때, 동시 멀티스레드 기능을 효과적으로 억제할 수 있는 상호배치(Interleave) 방식으로 냉각 루프가 각 하드웨어-스레드에서 가동된다. 동시 멀티스레드는 중첩 캐시 미스(Overlapping cache misses)와 파이프라인 스톨(Pipeline stalls)을 통하여 코어의 활용을 증가시키기 때문에, 동시 멀티스레드 기능을 억제하면 코어 활용이 줄고, 따라서 코어의 작동 전력과 온도도 줄어든다. 그림 8은 싱글스레드 시스템에서 가동되고 있는 냉각 루프를 보여준다. 여기서는 냉각을 위해 사용자 태스크가 일시적으로 정지된다.

4. 관련 연구

많은 연구자들이 소프트웨어와 하드웨어를 이용하여 시간과 공간의 다양한 정밀도수준에서 온-칩 동적 열관리(DTM)개념을 다루었다. 하드웨어 기술들[3,7]은 프로세서내의 단위 요소에 사용되는 전략을 재분배함으로써 국지적인 핫 스폿을 효과적으로 제한 할 수 있다는 것을 보이고 있는데, [3]은 단일 코어(Single-core) 프로세서의 온도를 동적 전압/주파수 스케일링(Dynamic Voltage Frequency Scaling), 인출 쓰로틀링, 스페큘에이션 제어(Speculation control) 및 다른 기술에 의해서 줄일 수 있음을 보여준다. POWER 5[4]는 온-칩 열 센서들이 위험 온도들을 탐지할 때 2-단계 인출 쓰로틀링를 이용한다. 이러한 하드웨어 기술들은 칩을 보호하는데 중요하다; 어떤 경우에는, 대부분의 애플리케이션들의 성능을 떨어뜨리지 않으면서 더 저렴한 패키지/냉각 솔루션을 가능케 한다. 일부 하드웨어 스로틀 기술들은 [6]에 나온 것과 같이 특정한 작업부하 시나리오에서 심한 성능 손상을 일으킬 수도 있지만, 이러한 동적 열관리기능은 오늘날 많은 상업용 프로세서들에서 이미 지원 되고 있는 기능이다[6].

칩에 있는 다수의 코어들에서 동시 멀티스레드를 활용하는 현대의 마이크로프로세서들은 태스크 스케쥴링, 운영체계와 시스템 하이퍼바이져와 같은 시스템-수준 소프트웨어에서의 스케쥴링등을 이용하는 새로운 열 관리 기법들을 제시한다. 운영체계에는 열 상수들보다 훨씬 더 작은 시간 정밀도를 다루는 태스크 스케쥴링이 이미 구현 되어 있기 때문에, 성능에 큰 영향을 주지 않으면서 공간적 및 시간적 열틈새를 조절하는 열 제어 기술들을 스케쥴링 루틴들에 포함 할 수 있다.

칩 멀티코어 프로세서의 코어들은 전형적으로 캐쉬의 적중율 향상을 위해 캐쉬들을 공유한다. 공유된 캐쉬들로 인하여 한 코어에서 다른 코어로의 태스크의 전이(Migration)패널티는 줄어 들고, 따라서 코어 반복이동을 실질적인 열 제어 기술이 되게 한다. [8,9]에서 우리는 다양한 코어 반복이동 기술을 볼 수 있다. [10]은 이종 칩 멀티코어 프로세서에서 시스템에서의 코어의 성능차이를 기반으로 코어 반복이동을 제시한다. 한편, 코어에서 실행하는 태스크들의 조합이 성능과[11,12] 온도에[13] 중요한 영향을 미치기 때문에, 온도-인식 동시 멀티스레드 공동 스케쥴링(Co-scheduling)은 실행가능한 또 다른 열 제어 기술이다. [14]에 보고된 연구는 열 시간 상수들이 운영체계 시간 구획들보다 더 클 경우, 태스크의 실행 순서를 적절리 조정함으로써 하드웨어기반의 패치 쓰로틀링을 촉발시키는 빈도를 줄여서 온-칩 온도를 낮추는 동시에 성능감소를 최소화 할 수 있음을 보여준다. 다른 연구들 [15]은 운영체계 스케쥴링과 동적 전압 주파수 스케일링과 같은 하드웨어 기술을 결합하여 운영체계 수준의 열 완화 기술을 향상 시킬 수 있음을 보여준다. 이들 연구와는 달리 우리는 [15]에서 사용된 온도 추정 계산법을 사용하지 않고 온-칩 마이크로 디지털 센서를 사용하여 실제 온도를 측정했다. [15]에서는 칩멀티프로세서 아키텍처가 열 문제들을 악화시키므로 효율적인 프로세서의 냉각을 위해서는 태스크를 칩 외부로 내보야 한다고 주장하지만, 우리는 운영체계가 가 칩 멀티코어 프로세서 칩 내에 존재하는 공간 및 시간적 열틈새를 활용 하여 성능에 큰 영향을 주지 않고 운영 온도를 크게 줄일 수 있음을 보였다.

5. 결 론

이 논문에서, 우리는 시스템의 성능이 빠른 속도로 증가하고 이에 따라 전력소비의 증가에 따른 발열에 의한 프로세서의 운영온도에 제한이 발생하는 문제를 해결하기위하여, 칩 멀티 코어 프로세서에 사용가능한 시간 및 공간적 핫 스폿 완화 설계와 열 시간 상수, 작업부하 변동과 마이크로프로세서의 전력 배분에서 발생하는 상충관계를 1.2GHz POWER 5 시스템에서 조사했다. POWER 5에 하드웨어 기반으로 구현된 앞선 클락 게이팅(Clock-gating)과 같은 전력관리 기술 때문에 태스크 배분과 작업부하 특성이 온-칩 온도에 중요한 영향을 주는 것을 보였다.

뿐만 아니라, 온-칩 온도 문제들을 완화시키기 위해서 운영체계-수준 열 인식 스케쥴링의 영향들도 조사했다. 코어 반복이동의 가능성을 실제 시스템에서 측정을 통하여 확인하였으며, 스케쥴링적용에 중요한 역할을 하는

반응 시간에 대한 조사도 실시하여 시스템 수준에서 구현하여도 충분히 효과적으로 대응 할 수 있음을 보였다.

먼저 고온 태스크의 지연 실행방안에서 태스크의 실행 순서를 조절함으로써 시스템의 최고 온도를 효과적으로 제어 할 수 있음을 보였다, 그리고 냉각 루프를 도입하여 동시 멀티스레드 및 단일스레드 환경에서 적용가능한 스케쥴러를 구현하였다. 우리의 방안은 공간 및 시간적 열틈새를 활용하여 운영체계의 협력과 이미 상용 프로세서에서 존재하고 있는 하드웨어를 이용하여 프로세서상에서 실행되고 있는 작업부하의 특성을 적시에 변경시킴으로써 온-칩 구성요소의 온도를 낮출 수 있게 한다.

우리의 방안은 운영 온도를 최대 5.5도씨까지, 평균 3.5도씨 낮추며, 이 때 1.08% 평균 성능 저하를 가진다. 미래의 고성능, 멀티-코어 시스템에서는 작업부하 변동과 보다 공격적인 전력 관리 기술들이 도입되어 이러한 시스템에서 이용 가능한 공간 및 시간적 열틈새가 늘어날 것이므로 우리가 더 큰 운영온도를 낮출 수 있을 것으로 기대된다. 열 인식 스케쥴링에 의한 실제 효과를 정량화하기 위해서 우리의 실험을 운영체계 수준에서 실시했으나, 우리는 시스템 하이퍼바져도 이 설계를 실행할 수 있다고 믿는다. 최근에 들어 많이 부각 되고 있는 가상 시스템 환경에서, 다수의 운영체계들의 상호작용과 조절을 통해서, 시스템 하이퍼바져도 이러한 온도 조절 효과를 이룰 수 있다.

참 고 문 헌

[1] L. Yeh and R. Chy. Thermal Management of Microelectronic Equipment. American Society of Mechanical Engineering, 2001.

[2] S. Gunther, F. Binns, D. Carmean, and J. Hall. Managing the Impact of Increasing Microprocessor Power Consumption. Intel Technology Journal, 5, February 2001.

[3] D. Brooks, M. Martonosi. Dynamic thermal management for high performance microprocessors. In Proceedings of the International Symposium on High Performance Computer Architecture (HPCA), January 2001.

[4] J. Clabes, J. Friedrich, M. Sweet, J. DiLullo, S. Chu, D. Plass, J. Dawson, P. Muench, L. Powell, M. Floyd, B. Sinharoy, M. Lee, M. Goulet, J. Wagoner, N. Schwartz, S. Runyon, G. Gorman, P. Restle, R. Kalla, J. McGill, and S. Dodson. Design and implementation of the power5 microprocessor. In Proceedings of the Design Automation Conference (DAC), 2004.

[5] H. F. Hamann, J. Lacey, A. Weger, and J. Wakil. Spatially-resolved imaging of microprocessor power (SIMP). In Proceedings of the Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems(ITherm), May 2006.

[6] R. Kotla, S. Ghiasi, T. Keller, and F. Rawson. Scheduling Processor Voltage and Frequency in Server and Cluster Systems. In Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS), April 2005.

[7] K. Skadron, M. Stan, W. Huang, and S. Velusamy. Temperature Aware Microarchitecture. In Proceedings of the International Symposium on Computer Architecture (ISCA), June 2003.

[8] J. Srinivasan and S. V. Adve. Predictive Dynamic Thermal Management for Multimedia Applications. In Proceedings of the International Conference on Supercomputing, June 2003.

[9] E. Kursun, G. Reinman, S. Sair, A. Shayesteh, and T. Sherwood. Low-Overhead Core Swapping for Thermal Management. In Proceedings of the Power-Aware Computer Systems Workshop, December 2004.

[10] S. Ghiasi and D. Grunwald. Thermal Management with Asymmetric Dual-Core Designs. Technical Report CU-CS-965-03, University of Colorado, 2004.

[11] A. Snavely, D. Tullsen, and G. Voelker. Symbiotic jobscheduling with priorities for a simultaneous multithreading processor. In Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), June 2002.

[12] D. M. Tullsen, S. J. Eggers, and H. M. Levy. Simultaneous multithreading: maximizing on-chip parallelism. In Proceedings of the International Symposium on Computer Architecture (ISCA), June 1995.

[13] M. D. Powell, M. Gomaa, and T. N. Vijaykumar. Heat and run: Leveraging smt and cmp to manage power density through the operating system. In Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XI), October 2004.

[14] E. Kursun, C-Y. Cher, A. Buyuktosunoglu, and P.Bose. Investigating the Effects of Task Scheduling on Thermal Behavior. In Proceedings of the Workshop on Temperature-Aware Computer Systems (TACS), June 2006.

[15] A. Merkel and F. Bellosa. Balancing Power Consumption in Multiprocessor Systems. In Proceedings of the ACM SIGOPS EuroSys Conference, April 2006.

최 정 환

한국과학기술원 전산학과에서 학사, 정보 및 통신공학과에서 석사, 전산학과에서 박사 학위를 1990, 1997, 2007년에 각각 수여 받았습니다. 현재 미국 펜실바니아 주립대에서 박사후 연구원으로 재직하고 있습니다. 중요 연구분야는 컴퓨터 시스템의 전력 관리, 시스템 버츄얼라이제이션, 온도인식 컴퓨팅이다.