9

최근 프로세서 발열 관리 연구 동향

고려대학교 ▌공준호 · 정성우*

1. 서 론

오늘날 마이크로프로세서 디자인 이슈에는 여러 가지 요소들이 있다. 디자인 시에 고려해야 하는 요소들 중 성능의 경우, 예전부터 가장 중요시되는 요소로서 성능은 가장 첫 번째로 고려되는 요소이다. 따라서 90년대 초반까지만 하더라도 연구자들은 다른 요소들보다는 성능을 높이는 데에 주안점을 두고 프로세서를 설계하였다. 그러나, 성능을 높이기 위해서 발전된 프로세스 기술을 사용하고, 칩 안에 들어가는 트랜지스터들의 집적도가 높아지면서, 전력 문제가 또 하나의 중요한 문제로 대두되었다. 전력 문제는 특히 모바일이나 임베디드 프로세서 설계에서 중요한 문제였는데 이유는 배터리 수명이 전력과 직접적인 연관이 있기 때문이다. 또한, 서버시스템의 경우 전력소모가 많아지면 전기료가 많아지고 이는 대형 서버를 유지해야 하는 기업 측면에서 보면 크나큰 손실이 아닐 수 없다. 전력 문제가 프로세서 설계 시 크나큰 장벽으로 대두되면서 더불어 발열 문제 또한 심각한 문제가 되었다. 프로세서의 발열량은 일반적으로 프로세서의 전력 소모량과 넓이, 두 가지 요소가 중요하게 작용한다. 전력소모량이 많을수록, 칩의 영역이 작을수록 프로세서의 온도가 높아진다. 즉, 프로세스 기술이 발전하면서, 전압 스케일링(Voltage scaling)으로 인해 전력소모량은 줄어들고 있지만, 트랜지스터의 크기는 더더욱 줄어들었기 때문에 칩의 크기가 작아지면서 온도 문제가 심각해졌다.

프로세서의 온도가 중요한 이유는 프로세서의 신뢰성에 직결되는 요소이기 때문이다[1]. 온도가 과도하게 높아지게 되면 최악의 경우 프로세서가 타버릴 가능성도 있다. 특히, 최근 연구에서는 악의적으로 한 부분의 온도를 올릴 수 있는 악성코드도 소개된 바 있기 때문에[2] 온도와 관련된 신뢰성 문제는 매우 중요하게 다뤄져야 한다. 또한, 프로세서가 온도가 높은 상태로 오래 유지될 경우 에이징(aging) 현상이 가속화되어 프로세서의 수명이 단축된다. 온도가 높아질 경우 또 한가지 문제는 트랜지스터의 스위칭 속도가 느려진다는 점이다. 트랜지스터 스위칭 속도가 느려지면 한 클럭 사이클 안에 수행되어야 하는 로직이 시간이 오래 걸려 한 사이클 이상 시간이 소요되는 경우도 있다. 이 경우, 프로세서의 오작동을 일으킬 소지가 많다.

그림 1은 프로세서의 발열이 얼마나 심각한 문제인가를 단적으로 보여주는 예이다[3]. 현재 프로세서의 발전 추세를 고려하였을 때 머지않은 미래에 단위 면적 당 전력 소모량이 원자로, 로켓 노즐, 그리고 결국 태양 표면의 수준까지 다다를 수 있다는 그림이다. 물론 이런 정도로 온도가 상승하게 프로세서를 놔둔다면 프로세서는 동작할 수 없을 것이다. 따라서, 프로세서의 발열문제를 효율적으로 해결하기 위해서 2000년대 이후 많은 연구들이 수행되어왔다.

본 논문은 프로세서의 발열 문제를 해결하기 위한 연구들(프로세서 온도의 모델링, 프로세서의 발열 관리 기법들)에 대해서 간략히 소개하고 향후 프로세서의 발열 연구에 대한 방향과 통찰력을 제시하는 것이 주 목적이다. 본 논문의 구성은 2장에서는 프로세서의 온도 모델링 기법에 대해 간략히 설명한다. 3장에서 최근 대표적인 프로세서 온도 관리 기법들에 대

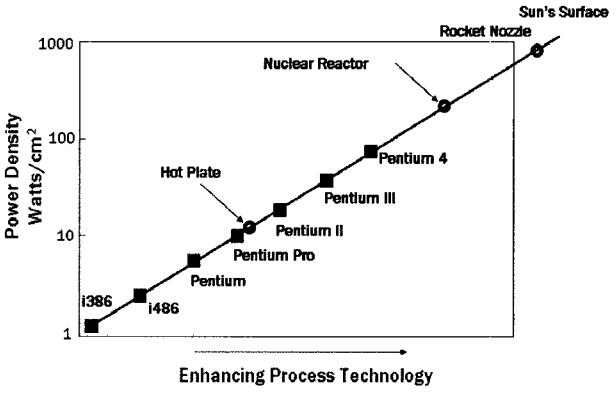

그림 1 프로세서의 발열 추세[3]

* 종신회원

해서 설명하고, 마지막으로 4장에서 논문의 결론을 도출한다.

2. 프로세서 온도 모델링

프로세서 설계 시 온도에 대한 모델링/시뮬레이션 프레임워크는 상당히 중요하다. 이는 제품의 온도 안정성을 테스트해볼 수 있을 뿐 아니라 제품의 time-to-market과 비용을 줄이는 데에도 중요한 역할을 한다. 본 논문에서는 제일 널리 쓰이고 있는 HotSpot 모델[1]과 프로세서의 런타임에 온도를 측정하기 위해 HotSpot에서 약간 변형된 성능계수기[4]를 이용한 모델, 그리고 여기서 발전된 단순 회귀분석(simple regression) 을 이용한 모델[6,7]에 대해 간략히 설명하고자 한다.

2.1 HotSpot 온도 모델

HotSpot 온도 모델은 현재 컴퓨터 구조 커뮤니티에서 가장 널리 쓰이고 있는 모델이다. HotSpot 온도 모델은 온도적 RC 모델에 기초하고 있으며, 프로세서 상의 기능 유닛 혹은 부분들을 전기 저항(resistance)과 전기 용량(capacitance)로 모델링하였다. 실리콘이 실제 칩 부분이고, 칩 안에는 여러 컴포넌트 혹은 블록들로(일반적으로 기능 유닛 별로) 나누어져 있다. Heat sink와 heat spreader도 같이 모델링 되어 있으며, 실리콘에서 발생한 열이 heat sink와 heat spreader를 거쳐 주변 공간(ambient)으로 나가게 된다. 표 1은 온도적 RC 모델과 실제 열역학에서 쓰이는 요소들을 비교한 표이다. RC 모델은 4차 Runge-Kutta method를 통해 해를 구할 수 있으며 표 1에서 보이는 바와 같이 전압을 온도 차이로 바꿀 수 있다. HotSpot은 인접성(adjacency)를 고려하기 위하여(인접성은 열이 전도되는 데에 매우 중요한 역할을 한다) 플로어플랜을 인풋으로 받는다. 플로어플랜은 각각의 기능 유닛들이 어느 위치에 배치되어 있는지를 나타내는 정보이다. 모델 검증 결과 상용 프로그램인 Floworks와 비교하여 최대 5.8%, 평균 3% 이하의 오차를 보여주는 것으로 나타났다.

표 1 온도적 RC 모델과 실제 열 역학 비교

온도적 관점(단위)	전기적 관점(단위)
열 흐름, 전력(W)	전류 흐름(A)
온도 차이(K)	전압(V)
온도 저항(K/W)	전기 저항(Ω)
열 용량(J/K)	전기 용량(F)
온도적 RC 상수(s)	전기적 RC 상수(s)

2.2 성능계수기를 이용한 온도 모델

성능계수기를 이용한 온도 모델[4]은 프로세서의 런타임에 동적으로 온도를 측정하는 모델로써 성능계수기를 통해서 각 기능 유닛의 접근 횟수를 추출한 뒤, 접근 횟수를 바탕으로 각 기능 유닛의 전력 소모량을 측정한다. 전력 소모량은 식 (1)과 같이 측정될 수 있다[5].

$$\text{Power} = \text{Access Rate} \times \text{Architectural Scaling} \times \text{Max Power} + \text{Non Gated Clock Power} \quad (1)$$

Access Rate는 성능계수기에서 추출한 접근 횟수이고, Architectural Scaling은 아키텍쳐 구조에 따라서 적절한 전력 소모량 값을 만들기 위해 달라지는 계수, Max Power는 접근 당 최대 파워 소모량, 그리고 Non Gated Clock Power는 클럭 파워 소모량이다. 각 기능 유닛 별 파워 소모량을 추출한 후에 HotSpot을 이용하여 온도를 계산한다. [4]에서는 Pentium 4를 이용한 모델이고 최근에는 Intel Core2 Duo를 모델로 한 연구도 있다[6].

그러나, HotSpot을 이용한 온도 모델은 행렬의 4차 미분방정식(Runge-Kutta Method)의 해를 구해야 하므로 계산의 오버헤드가 심해서 실제 런타임에 온도를 측정하는데 사용되기 힘들다. 따라서, 더 간단한 모델이 제안되었다. 단순 회귀분석(Simple Regression)을 이용한 모델[6,7]인데, 이 모델은 칩 제조시 온도 시뮬레이션 결과를 통하여 기능 유닛의 접근 횟수와 온도와의 관계를 식 (2)에서 보이는 바와 같이 1차 함수로 바꾼다.

$$\text{Temperature} = a * \text{Access Rate} + b \quad (2)$$

'a'와 'b'값은 칩 제조시 온도 시뮬레이션을 통해 산출 가능하다. 접근 횟수(Access Rate)만 있으면 일반 HotSpot에서 사용하는 4차 미분방정식을 사용하는 것보다 훨씬 간단하게 온도를 계산할 수 있다. 이는 실제로 HotSpot 모델이 온라인(런타임) 온도 측정에 사

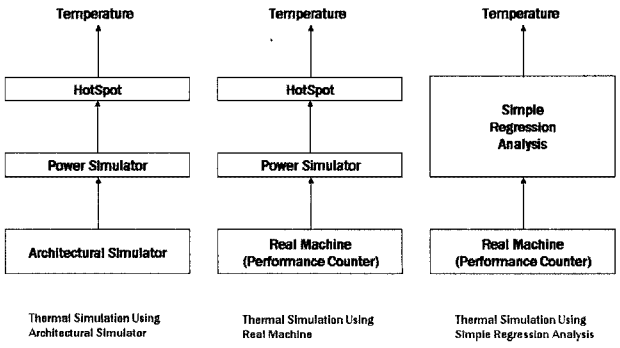

그림 2 여러 온도 시뮬레이션 기법들[7]

용되기 힘들 때 유용하다. 독자들의 이해를 돕기 위하여, 그림 2는 여러 가지 프로세서 온도 시뮬레이션 기법들에 대해서 그림으로 보여주고 있다. 위에서 설명했던 일반 HotSpot을 사용한 시뮬레이션, 프로세서의 성능 계수기를 이용한 시뮬레이션, 그리고 단순 회귀분석을 이용한 시뮬레이션을 비교한 그림이다.

3. 프로세서 발열 관리 기법들

여러 계층에서 프로세서의 발열 문제를 해결하기 위해 많은 기법들이 제안되어 왔다. 본 논문에서는 이제까지 제안되었던 대표적인 프로세서 온도 관리 기법들에 대해서 계층적으로 접근하여 독자들의 이해를 돕고자 한다. 그림 3은 온도 관리 기법 연구들의 계층 구조에 대해 보여주고 있다. 본 논문에서는 프로세서 온도 관리 기법에 대해서 크게 정적 기법과 동적 기법으로 나누어 접근하고자 한다. 정적 기법은 프로세서의 런타임에 온도를 유연하게 조정하는 것이 아닌, 프로세서의 디자인 시에 온도를 고려한 설계를 하는 것이다. 이에 반해, 동적 기법은 프로세서의 런타임에 온도를 감지하여, 이에 맞는 온도 관리 기법을 적용하는 것을 말한다. 정적 기법은 다시 프로세서의 구조 개선, 플로어플래닝, 그리고 컴파일러 기반 기법으로 나누어진다. 프로세서 구조 개선은 발열에 강한 프로세서 구조를 설계하는 것이고, 플로어플래닝은 프로세서의 기능 유닛(Functional Unit) 들의 위치를 정하는 것이다. 컴파일러 기반 기법은 프로그램의 코드를 발열이 적게끔 최적화하는 것으로 런타임 시에 수행되는 것이 아니기 때문에 정적 기법으로 분류하였다. 동적 기법에 대한 연구는 하드웨어 기반 기법과 소프트웨어 기반 기법으로 나누어 설명하고자 한다.

3.1 정적 기법

3.1.1 프로세서 구조의 개선

정적 기법 중 프로세서 구조의 개선은 프로세서 디자인 시에 구조를 발열에 강하게끔 설계하는 기법이다. 초기 연구로 이중 파이프라인 구조는[8] 프로세서가 발열이 심해질 경우를 대비해 두 벌의 파이프라인을 각각 운영한다. 평시에는 Out-of-order 파이프라인을 사용하다가 프로세서의 발열이 심해지면 out-of-order 파이프라인보다 파워소모가 적은 in-order 파이프라인을 사용한다. 기존의 out-of-order 파이프라인은 다시 사용되기 전까지 클럭 게이팅(Clock Gating) 된다. 다른 연구로 activity migration 기법이 있는데 이는 프로세서의 기능 유닛을 여러 벌 두고, 한 곳의 기능 유닛의 사용이 많아지면 다른 벌의 기능 유닛으로 실행을 전환한다[11]. 그림 4는 이러한 activity migration을 지원하기 위한 프로세서 구조를 예를 들어 보여주는 그림이다. 예시 1(Example 1)에서는 rename unit/issue queue와 register file과 execution unit이 중복 되어 발열이 심할 경우 중복된(replicated) 기능 유닛으로 실행을 전환할 수 있다. 예시 2(Example 2)에서는 예시 1에 덧붙여 instruction fetch unit이 중복되어 있다. 프로세서 디자인 시에 어느 기능 유닛이 두 벌 만들어져야 하는지는 프로세서 디자이너가 온도 시뮬레이션을 통해 정할 수 있다. 이와 비슷한 기법으로 migrating computation 기법이 있다[1]. 실제 프로세서에서 정수 레지스터 파일이 가장 뜨거운 점을 이용하여, 두 벌의 정수 레지스터 파일을 두고, 한쪽의 레지스터 파일이 뜨거워지면, 레지스터 파일의 값들을 복사하여 두 번째 레지스터 파일에 넣고, 두 번째 레지스터 파일을 사용한다. 최근 구조 개선 연구로는 banked 레지스터 파일을 이용한 구조가 있다[9]. 이 기법은 정수 레지스터 파일이 일반적으로 프로세서에서 가장 뜨겁다는 사실에 착안하여 banked 레지스터 파일 구조를 적용, 레지스터 파일의 온도를 낮추고 동적 온도 관리(DTM)이 일어나는 횟수를 줄여, 성능 향상에도 기여한다. 이러한 컴퓨터 구조적인 접근 외에 다른 방향에서 접근한 연구로는 최근 3D 프로세서에서 온도문제가 심각한 점을 이용, 3D 프로세서에서 물을 이용한 프로세서 과열방지 기법(liquid cooling)과 이러한 기법이 프로세서에 미치는 영향에 대해서 분석한 연구이다[10]. 물을 이용한 과열방지 기

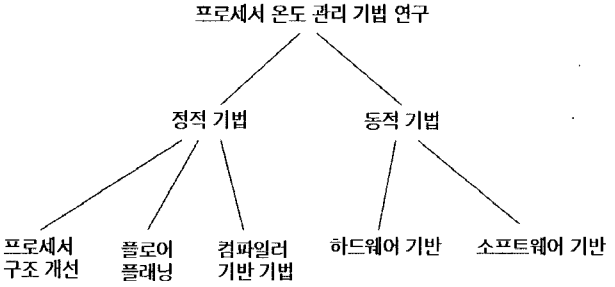

그림 3 온도 관리 기법의 분류

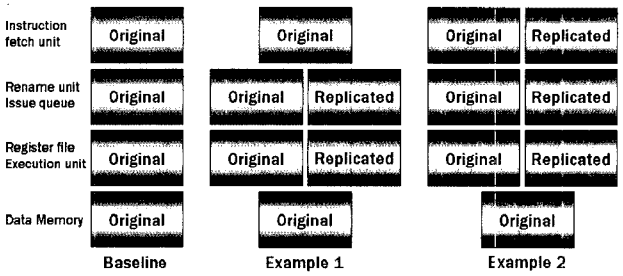

그림 4 Activity Migration을 위한 하드웨어 구조

법을 사용하였을 경우, 줄어든 DTM 횟수 때문에 성능 향상에 기여하고, 온도 저하로 인하여 프로세서의 신뢰성에도 좋은 영향을 준다.

3.1.2 플로어플래닝(Floorplanning)

플로어플래닝도 프로세서의 디자인 시에 프로세서의 기능 유닛들의 위치를 결정하는 것으로 발열과 상당히 깊은 연관성이 있다. 프로세서의 발열을 고려한 플로어플래닝 시에 중요한 점은 발열이 심한 기능 유닛들을 가깝게 배치하지 않는 것이다. 이유는 발열이 심한 유닛들이 서로 가까운 위치에 있으면 열 전도 상호작용에 의해서 서로 더 뜨거워지는 현상이 발생한다. 따라서, 뜨거운 기능 유닛들을 가급적 멀리 배치하는 것이 발열을 고려한 플로어플래닝의 기본이다.

플로어플래닝은 컴퓨터 구조적인 접근 보다는 CAD (Computer-Aided Design) 관점의 접근이 일반적이다. 연구자들이 집중적으로 연구해왔던 분야도 칩 디자인 단계에서 플로어플래닝 알고리즘에 집중되어 있다. 널리 알려진 온도 시뮬레이션 툴인 HotSpot[1]에 포함되어 있는 HotFloorplan 툴도 CAD 관점의 플로어플래닝 알고리즘(simulated annealing 알고리즘)을 적용한 툴이다[12]. 이 툴에서 주안점을 두고 있는 세 가지의 요소는 칩 영역의 넓이, 온도, 그리고 선 길이(wire length)이다. 온도를 고려한 플로어플래닝 툴(tool) 이다보니 온도를 고려하는 것은 당연하고, 선 길이는 성능에 밀접한 연관이 있기 때문에 고려된다. 칩 설계자가 칩 영역의 넓이 혹은 선 길이에 가중치를 줄 수 있어서 어느 요소가 더 중점적으로 고려되어야 하는지에 대한 유연성을 제공한다. 이와 비슷한 알고리즘으로 Han et al의 기법[13]이 있는데 칩 영역의 넓이, 온도, 선 길이 세 가지를 고려한다는 점과 simulated annealing 방법을 사용한다는 점은 HotFloorplan[12]과 동일하지만 열이 전도되는 특징을 고려하여 인접한 기능 유닛 사이의 열전도량을 추가적으로 고려한다. 위에서 소개된 두 가지 기법과는 다른 관점으로 접근된 기법으로는 CPI(Clock cycle Per Instruction)를 고려한 것인데 위의 두 기법도 선 길이로 성능을 고려하긴 하지만 선 길이만으로는 정확한 성능을 반영하기 힘들다. 따라서, 성능의 직접적인 척도가 되는 CPI를 고려한 플로어플래닝 알고리즘이 제안되었다. Nookala et al이 제안한 알고리즘[14]은 CPI를 추가적으로 고려하여 HotFloorplan보다 온도와 성능 측면에서 뛰어난 결과를 보여주고 있다. 이와 비슷한 연구로 Chu et al의 연구[15]가 있는데 이도 비슷하게 온도와 칩 영역의 넓이, 그리고 CPI를 동시에 고려하는 알고리

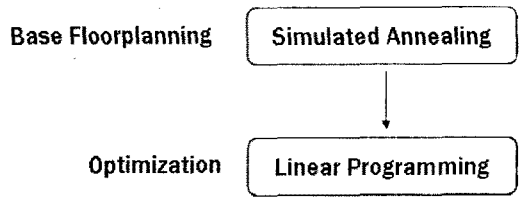

그림 5 Simulated Annealing (SA) + Linear Programming (LP) 플로어플래닝

즘이다. 플로어플래닝 알고리즘에서 simulated annealing 방식이 많이 쓰이지만, simulated annealing(SA)과 linear programming(LP) 방식을 혼용한 알고리즘도 소개되었다[16]. 이 알고리즘에서는 그림 5에서 보이는 것과 같이 LP를 이용하여 기초적인 플로어플레닝을 수행한 뒤, SA로 최적점에 가깝게 접근하는 방식을 사용하였다. 결과로 SA나 LP 단독으로 플로어플래닝을 하는 것보다 SA와 LP를 동시에 사용하는 것이 온도나 성능 측면에서 더 좋은 결과를 가져다주었다.

3.1.3 컴파일러 기반 기법

하드웨어 설계 최적화 관점에서가 아닌 소프트웨어 계층에서도 온도 최적화가 가능하다. 컴파일러 계층에서 이러한 코드 최적화를 수행할 수 있다. 컴파일러 계층의 최적화가 좋은 이유는 추가적인 하드웨어의 오버헤드가 없다는 점이다. 하드웨어 복잡도를 줄임으로써 추가적인 전력 소모 또한 막을 수 있다. 또한, 컴파일러 계층의 과열 방지 기법은 하드웨어가 동적으로 막지 못하는 과열 상황을 막아주는 guard band 역할도 동시에 수행 가능하며, 하드웨어의 짐을 덜어주는 효과도 가지고 있다.

컴파일러 계층의 온도 최적화는 VLIW(Very Long Instruction Word) 구조를 타겟으로 주로 연구되어 왔다. VLIW 구조는 명령어 스케줄링이 전적으로 컴파일러에 의존하므로 컴파일러 계층에서의 온도 최적화가 매우 중요하다. Mutyam et al의 VLIW 구조 컴파일러 계층 최적화는 프로세서 상에서 기능 유닛의 사용을 한쪽으로 치우치지 않고 균형되게 함으로써 특정 부분의 과열을 막는 구조이다(Load balancing)[17]. 또한, 사용되지 않는 기능 유닛들의 전력 공급을 막음으로써 온도를 줄인다. Schafer et al에 의해 제안된 기법[18]은 프로세서의 과열이 예상되는 지점에 컴파일러가 NOP (No-operation) 명령어를 중간에 삽입함으로써 과열을 막는다. NOP 명령어는 명령어 반입 이외에 아무 역할도 수행하지 않기 때문에 온도를 낮출 수 있다.

3.2 동적 온도 관리(Dynamic Thermal Management)

여러 계층에서 동적 온도관리가 가능하다. 낮게는

회로(circuit) 계층부터 높게는 시스템소프트웨어/운영체제 계층까지 다양한 계층에서 동적 온도관리 기법을 적용 가능하다. 본 논문에서는 하드웨어 기반 기법과 소프트웨어 기반 기법으로 나누어 접근 한다.

3.2.1 하드웨어 기반 기법

동적인 온도 관리 기법(DTM: Dynamic Thermal Management)은 동적으로 프로세서의 온도를 감지하여 프로세서의 온도가 일정 온도 이상 상승하게 되면 프로세서의 전력 소모를 줄이는 저전력 모드로 들어가게 된다. 저전력 모드라 함은 프로세서를 잠깐 쉬게 하여 열을 식히는 방법(클럭 게이팅 또는 파워 게이팅) 또는 DVFS(Dynamic Voltage & Frequency Scaling), IPC throttling(컴퓨터 구조적으로 issue width 등 명령어 처리의 대역폭을 동적으로 조절하는 기법) 등이 많이 사용된다. 그러나, 프로세서를 쉬게 하는 것, 클럭 주파수를 낮추는 것, 또는 명령어 처리 대역폭을 줄이는 것은 그만큼 성능 저하를 야기한다. 최근 연구자들은 이러한 DTM 기법을 적용할 때 성능의 저하를 최소화하는 방법에 대한 연구에 주력해왔다. DTM 초기 연구로 Brooks et al의 연구는 DTM이 성능에 미치는 영향에 대해서 연구하였다[19]. 원래 열이라는 것은 전도되는데 일정 시간이 소요되기 때문에 DTM 기법 적용 시 온도가 바로 내려가는 것이 아니고, 시간이 소요된다. 따라서, 이러한 시간들을 적절히 고려해야 최적의 성능을 보이는 DTM 기법을 고안할 수 있다.

DTM 하에서 성능을 높이기 위하여 여러 다른 DTM 기법이 제안되어왔는데, hierarchical DTM과 hybrid DTM기법이다. Hierarchical DTM[20]은 프로세서의 온도에 따라서 계층적인 DTM 기법을 적용하는 것으로 프로세서의 온도가 올라감에 따라서 처음에는 filter cache같은 많은 성능저하를 유발하지 않는 기법을 적용하고, 프로세서의 온도가 높아졌을 때 클럭 게이팅(clock gating) 같은 온도를 많이 낮출 수 있지만, 많은 성능 저하를 유발하는 기법을 적용한다. Hybrid

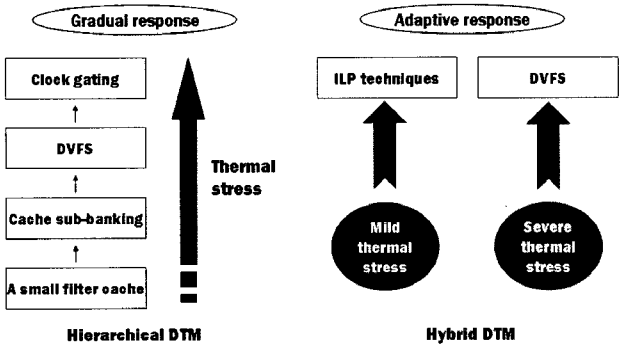

그림 6 Hierarchical DTM과 Hybrid DTM

DTM 기법[21]은 hierarchical DTM 기법과 비슷하지만, 프로세서의 온도에 따라 유연한 DTM 기법을 적용하는 점이 다르다. 프로세서 상의 thermal stress의 정도를 감지하여 stress가 심하지 않을 때는 IPC throttling 기법을 적용하고, stress가 심할 경우 DVFS같은 강력한 DTM 기법을 적용하게 된다. 그림 6은 hierarchical DTM과 hybrid DTM의 차이를 보여준다. Control-theoretic DTM 기법[22]은 성능에 대한 피드백을 받고, 그 피드백의 결과에 따라 DTM 기법의 정도를 달리한다. PID (Proportional-Integral-Differential) 컨트롤러를 이용한 이 기법은 일반적인 DTM 기법에 비해 성능저하를 65%나 줄이는 효과가 있다.

일반적으로 프로세서 코어 이외에도 캐쉬 메모리의 과열을 방지하기 위한 기법들도 많이 소개되었다. 캐쉬 메모리의 경우, 1장에서 소개된 바와 같이 악의적인 코드에 의해 공격받을 가능성도 있으므로, 더욱 더 세심한 설계가 필요하다. L1 데이터 캐쉬에서 특정부분이 집중적으로 접근될 경우 캐쉬메모리도 과열될 수 있다는 점에 착안하여 John et al[23]은 separated subarraying 기법과 interleaved subarraying 기법을 제안하였다. 두 기법은 캐쉬메모리의 특정 부분에 집중되는 전력 밀도(power density)를 분산시키기 위하여 연속적인 주소가 맵핑되는 캐쉬메모리 부분들이 서로 거리가 멀게끔 떨어트려 놓음으로서 특정 부분에 집중되는 접근에 대한 과열을 막는 메커니즘이다. Ku et al[24]의 power density minimized cache와 block permutation scheme도 캐쉬 메모리의 온도를 관리하기 위한 기법들인데 위에서 소개된 John et al의 기법[23]과 비슷하게 연속된 주소를 갖는 캐쉬메모리의 물리적 위치를 바꿈으로써 전력 밀도를 줄이는 방법이다.

3.2.2 소프트웨어 기반 기법

소프트웨어 기반 기법은 운영체제(OS) 또는 시스템 소프트웨어 레벨에서의 프로세서 온도 관리를 지칭한다. 운영체제 계층에서의 프로세서 온도 관리는 프로세스(process) 혹은 태스크(task) 스케쥴링 기법이 대부분이다. 이제까지의 연구들을 살펴보면, 발열에 강한 태스크 스케쥴링 기법들의 기본 철학은 성능 오버헤드를 최소한으로 줄이면서 뜨거운 태스크와 차가운 태스크를 섞음으로써 프로세서의 과열을 막는 것이다. 따라서, 기법의 우월성을 증명하는 기준은 크게 1. 프로세서의 과열 방지 와 2. 성능 이렇게 두 가지 요소가 된다.

최근 소프트웨어 기반 기법에 대한 연구들은 멀티

코어 혹은 SMT(Simultaneous Multi-Threading)를 타겟으로 한 연구들이 많았다. 이는 태스크 스케쥴링에 있어서 시간적 슬랙(temporal slack) 뿐 아니라 공간적 슬랙(spatial slack)도 같이 활용하여 성능 오버헤드를 최소화하면서 발열 관리를 할 수 있기 때문이다. Powell et al[25]의 Heat-and-Run 기법은 쓰레드(thread)들을 모아서 한 코어를 뜨겁게 한 뒤, idle한 코어로 옮기는 방식이다. 다른 기법으로 뜨거운 태스크들은 우선 순위를 낮게 두고, 차가운 태스크들은 우선 순위를 높게 줘서 스케쥴링하는 기법이 제안되었다[26,27]. 이 기법은 thermal emergency 상황이 오면 하드웨어에서 지원하는 클럭 게이팅 등의 기법을 이용하여 소프트웨어와 하드웨어의 협업을 통한 발열 관리를 제공한다. Choi et al[28]의 Heat balancing, Deferred Execution of Hot Jobs 등도 멀티코어 혹은 멀티 스레딩 프로세서에서 존재하는 공간적 슬랙을 이용한 기법이라고 볼 수 있다. Choi et al이 제안한 또 하나의 특이한 온도관리 기법은 cool loop를 삽입하는 것이다. Cool loop란 loop안에서 아무 일도 하지 않는 코드 시퀀스를 지칭하며 cool loop를 실행시키면 프로세서는 아무 일도 하지 않는 효과를 발휘하게 되어 프로세서의 과열을 막을 수 있다. Yang et al[29]이 제안한 태스크 스케쥴링 기법이 다른 기법들과 틀린 점은 태스크의 순서를 고려했다는 점이다. 그림 7에서 보이는 바와 같이, 이 기법에서 중요시 여기는 점은 뜨거운 태스크와 차가운 태스크의 실행 순서를 고려했을 때, 온도 측면에서 뜨거운 태스크를 먼저 실행시키고 차가운 태스크를 나중에 실행하는 것이 반대의 경우보다 온도가 훨씬 낮게 나온다는 점이다. 이 기법을 이용하여 적은 DTM trigger를 유발하고 이는 3.25~4.7%의 성능 향상으로 이어졌다. Merkel et al은 task activity vector라는 소프트웨어 기반 발열 관리를 위한 새로운 metric을 제안하였다[30]. 운영체제는 응용의 task activity vector를 보고 프로세스들의 대기 큐(waiting queue)를 정렬하여 태스크 스케쥴링을 한다. 프로세서의 과열을 방지하기 위해서 인접하여 스케쥴링 되는 프로세스들의 task activity vector의 내적은 0에 가깝도록 스케쥴링 된다.

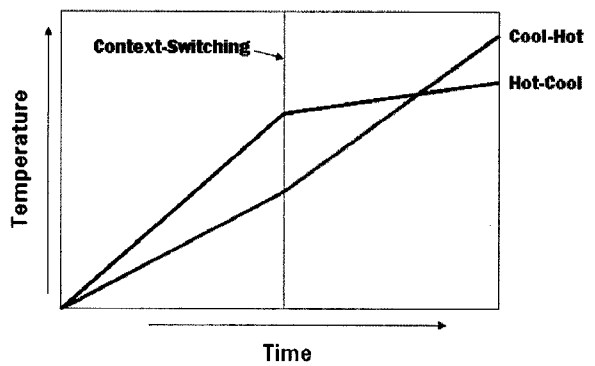

그림 7 태스크의 실행 순서에 따른 온도 변화

4. 결 론

이제까지 대표적인 프로세서 온도관리 기법들에 대해서 알아보았다. 본 논문에서는 프로세서 상에서 온도를 고려한 기법들을 특정 범주로 나누어 설명했지만, 여러 계층에서 복합적인 온도관리 메커니즘이 중요하다. 사실, 절대적으로 완벽한 온도 관리 메커니즘은 없기 때문에 여러 계층에서 메커니즘들의 보완적 상호작용이 매우 중요하다.

프로세서의 온도문제는 이제 프로세서 디자인 시에 절대적으로 고려해야 할 요소로 인식되고 있다. 특히, 온도는 프로세서의 신뢰성, 성능, 전력 소모 등 여러 면에 영향을 주기 때문에 더욱 더 중요하다. 3D 프로세서나 매니코어 같은 새롭게 주목받고 있는 기술들에도 언제나 온도 문제는 제 1순위로 고려되어야 하는 문제이다. 본 논문이 프로세서에서의 온도 관리의 중요성과 더불어 향후 연구 방향에 대한 통찰력을 제시할 수 있을 것이다.

참고문헌

[1] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, 2003. Temperature-Aware Microarchitecture. In *Proceedings of International Symposium on Computer Architecture (ISCA '03)*, 2003.

[2] J. Kong, J. John, E.-Y. Chung, Sung Woo Chung, and J. Hu, "On the Thermal Attack in Instruction Caches", accepted to IEEE Transactions on Dependable and Secure Computing.

[3] F. Pollack. New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies. *International Symposium on Microarchitecture (MICRO-32) keynote speech*, 1999.

[4] K.-J. Lee, K. Skadron, "Using Performance Counters for Runtime Temperature Sensing in High-Performance Processors," In *Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, 2005.

[5] C. Isci, M. Martonosi, "Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data," In *Proceedings of 36th Annual IEEE/ACM*

International Symposium on Microarchitecture(MICRO' 03), 2003.

[6] J. S. Lee, K. Skadron, S. W. Chung, "Predictive Temperature-Aware DVFS," *IEEE Transactions on Computers (On-line Publication)*, Sept. 2009.

[7] Sung Woo Chung and Skadron, K. "Using On-Chip Event Counters For High-Resolution, Real-Time Temperature Measurement," In *Proceedings of Thermal and Thermomechanical Phenomena in Electronics Systems, (ITHERM '06)*, 2006.

[8] C. H. Lim, W. Robert Daasch, and G. Cai, "A Thermal-Aware Superscalar Microprocessor," In *Proceedings of International Symposium on Quality Electronic Design (ISQED '02)*, pp. 517-522, 2002.

[9] H. B. Jang, E.-Y. Chung, and Sung Woo Chung, "Adopting the Banked Register File Scheme for Better Performance and Less Leakage", ETRI Journal, vol. 30, no. 4, pp. 624-626, August 2008.

[10] H. B. Jang, I. Yoon, C. H. Kim, S. Shin, and Sung Woo Chung, "The Impact of Liquid Cooling on 3D Multi-Core Processors", In *proceedings of IEEE International Conference on Computer Design (ICCD 2009)*, October 2009.

[11] S. Heo, K. Barr, and K. Asanović. "Reducing power density through activity migration," In *Proceedings of the 2003 international symposium on Low power electronics and design (ISLPED '03)*, pp. 217-222, 2003.

[12] K. Sankaranarayanan, S. Velusamy, M. R. Stan, and K. Skadron, "A Case for Thermal-Aware Floorplanning at the Microarchitectural Level," *The Journal of Instruction-Level Parallelism*, vol. 7, Oct. 2005.

[13] Y. Han and I. Koren, "Simulated Annealing Based Temperature Aware Floorplanning," *The Journal of Low Power Electronics*, vol. 3, No. 2, 1-15, 2007.

[14] V. Nookala, D. J. Lilja, and S. S. Sapatnekar, "Temperature-Aware Floorplanning of Microarchitecture Blocks with IPC-Power Dependence Modeling and Transient Analysis," In *Proceedings of the 2006 international symposium on Low power electronics and design (ISLPED '06)*, pp. 298-303, 2006.

[15] C-. T. Chu, X. Zhang, L. He, and T. T. Jing, "Temperature Aware Microprocessor Floorplanning Considering Application Dependent Power Load," In *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD 2007)*, pp. 586-589, 2007.

[16] M. B. Healy, M. Vittes, M. Ekpanyapong, C. S. Ballapuram, S. K. Lim, H-. H. S. Lee, G. H. Loh, "Multiobjective Microarchitectural Floorplanning for 2-D and 3-D ICs," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vol. 26, No. 1, pp. 38-52, 2007.

[17] M. Mutyam, F. Li, N. Vijaykrishnan, M. T. Kandemir, and M. J. Irwin, "Compiler-directed thermal management for VLIW functional units", In *Proceedings of ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES 2006)*, pp. 163-172, 2006.

[18] B. C. Schafer and T. Kim, "Thermal-Aware Instruction Assignment for VLIW Processors," In *Proceedings of 11th Workshop on Interaction between Compilers and Computer Architectures (INTERACT-11)*, pp. 1-7, 2007.

[19] D. Brooks, M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors," In *Proceedings of International Symposium on High-Performance Computer Architecture (HPCA '01)*, 2001.

[20] M. Huang, J. Renau, S-. M. Yoo, and J. Torrellas, "A framework for Dynamic Energy Efficiency and Temperature Management," *In Proceedings of International Symposium on Microarchitecture (MICRO 2000)*, 2000.

[21] K. Skadron, "Hybrid Architectural Dynamic Thermal Management," In Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '04), Vol. 1, 2004.

[22] K. Skadron, T. Abdelzaher, and M. R. Stan, "Control-Theoretic Techniques and Thermal-RC modeling for Accurate and Localized Dynamic Thermal Management," In *Proceedings of International Symposium on High-Performance Computer Architecture (HPCA '02)*, 2002.

[23] J. K. John, J. S. Hu, and S. G. Ziavras, "Optimizing the Thermal Behavior of Subarrayed Data Caches," In *Proceedings of International Conference on Computer Design (ICCD 2005)*, 2005.

[24] J. C. Ku, S. Ozdemir, G. Memik, and Y. Ismail, "Thermal Management of On-chip Caches Through Power Density Minimization," In *Proceedings of International Symposium on Microarchitecture (MICRO*

2005), 2005.

[25] 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'04)*, pp. 260-270, 2004.

[26] A. Kumar, L. Shang, L-. S. Peh, and N. K. Jha, "HybDTM: A Coordinated Hardware-Software Approach for Dynamic Thermal Management," In *Proceedings of the 43rd annual Design Automation Conference 2006 (DAC '06)*, pp. 548-553, 2006.

[27] A. Kumar, L. Shang, L-. S. Peh, and N. K. Jha, "System-Level Dynamic Thermal Management for High-Performance Microprocessors," IEEE transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 27, No. 1, pp. 96-108, 2008.

[28] J. Choi, C. Cher, H. Franke, H. Haman, A. Weger, and P. Bose, "Thermal-aware Task Scheduling at the System Software Level," In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED' 07)*, 2007.

[29] J. Yang, X. Zhou, M. Chrobak, Y. Zhang, and L. Jin, "Dynamic Thermal Management through Task Scheduling," In *Proccedings of the IEEE International Symposium on Performance Analysis of Systems and software (ISPASS '08)*, pp. 191-201, 2008.

[30] A. Merkel and F. Bellosa, "Task activity vectors: A new metric for temperature-aware scheduling," In *proceedings of Third ACM SIGOPS EuroSys Conference*, 2008.

공 준 호

2007 고려대학교 정보통신대학 컴퓨터과학 학사
2009 고려대학교 정보통신대학 컴퓨터전파통신공학과 석사
2009~현재 고려대학교 정보통신대학 컴퓨터전파통신공학과 박사과정 재학 중
관심분야 : 컴퓨터 구조, 고성능 컴퓨터, 임베디드 프로세서, 프로세서 전력/온도 관리
E-mail : luisfigo77@korea.ac.kr

정 성 우

2003 서울대학교 전기 컴퓨터공학 박사
2003~2005 삼성전자 반도체총괄, Senior engineer
2005~2006 방문 연구원, Department of Computer Science, University of Virginia
2006~현재 고려대학교 컴퓨터통신공학부 조교수
고려대 BK소프트웨어 사업단
관심분야 : 컴퓨터 구조, 프로세서 온도 관리, 플래쉬 메모리, System on Chip
E-mail : swchung@korea.ac.kr