# 에너지 효율을 위한 멀티코어의 동적 온칩캐쉬 스케일링

조정욱 ○ 서의성
울산과학기술대학교 전기전자컴퓨터공학부
{jmanbal, euiseong}@unist.ac.kr

# Dynamic On-chip Cache Scaling in Multicore Systems for Energy-Efficiency

Jungwook Cho○ Euiseong Seo
Dept, of Electrical and Computer Engineering, Ulsan National Institute of Science and Technology

**1. 서론** 집적되는 코어의 증가에 따라 멀티코어 구조에서는 온칩캐쉬의 크기가 증가하고 있으며, 이로 인해 온칩 캐쉬가 프로세서에서 소모하는 전력에서 차지하는 비중 역시 빠르게 증가하고 있다. 하지만, 실제 온칩캐쉬의 사용률은 실행되는 프로그램의 특성에 따라 상당한 변화를 보이므로 대용량 캐쉬를 필요로 하지 않는 경우 온칩캐쉬의 소모 전력은 불필요한 에너지 낭비로 이어지게 된다. 이런 문제를 해결하기 위해 동적 온칩캐쉬 스케일링은 워크로드마다 요구하는 메모리 사용 패턴이 모두 다르다는 사실을 활용하여 동적으로 온칩캐쉬의 구조와 용량을 변화시킴으로써 소비되는 전력을 줄이는 것을 목표로 한다. 본 논문에서는 기존의 온칩캐쉬 스케일링 방법의 한계점을 분석하고 멀티코어환경에서의 코어들 사이에서 공유되는 L2 캐쉬의 구조 및 크기를 변화시키는 알고리즘을 제안한다.

**2. 관련 연구** 동적 온칩캐쉬 스케일링에 대한 연구들은 크게 재구성이 가능한 온칩캐쉬 구조에 대한 연구와 [1,2,3] 태스크들의 특성을 분석하여 적절한 형태의 캐쉬 구조를 제공하는 알고리즘으로 [4,5] 구별할 수 있다. Albonesi는[1] 셋-어소시에티브 캐쉬(set-associative cache)에서 웨이(way)의 크기를 변화시킴으로써 캐쉬 크기를 스케일링하는 선택적인 웨이 캐쉬 구조를 논문에서 제시하였다. Yang은[2] 인스트럭션 캐쉬(instruction cache)에서 누설 에너지를 줄이기 위해 동적으로 캐쉬의 집합(sets)을 변화시키는 캐쉬 구조를 제시하였다. 그 후 동적 웨이 변화와 동적 집합 변화에 대한 효율성 비교가 Yang[3]에 의해 연구되었고, 두 가지 캐쉬 구조를 조합한 하이브리드(hybrid) 기법을 제안하였다. 효과적인 캐쉬 스케일링을 위한 알고리즘으로는 캐쉬 미스와 브랜치(branch) 명령어의 수를 이용하여 프로그램의 페이지 변화(phase change)를 감지하고 튜닝(tuning) 절차를 통해 해당 태스크의 특성에 최적화된 캐쉬 구조를 제공하는 Rochester 알고리즘[4]이 제안되었다. 또, 워킹 셋 시그니쳐(working set signatures)를 이용하여 실행 중인 프로그램의 워킹 셋(working set)의 변화를 감지하여 튜닝을 통해 재구성 가능한 하드웨어를 변화시키는 알고리즘[5]이 제안되었다. 위에서 제시된 연구에서는 싱글코어에서 단일 워크로드들의 특성만을 고려하였기 때문에 다수의 워크로드들이 동시에 실행되는 멀티코어 환경에 직접 활용하기가 어려우며, 성능에 대한 검증 역시 이루어지지 않았다.

**3. 멀티코어에서 캐쉬 스케일링 알고리즘** 캐쉬 스케일링 알고리즘 구현시 동적으로 변화하는 캐쉬 미스율이나 브랜치 연산의 빈도를 바탕으로 캐쉬 스케일링을 하는 것은 태스크 성능의 지나친 단순화를 통해 잘못된 결정을 유도할 수 있다. 따라서, 현재 시스템의 성능을 가장 정확하게 표현할 수 있는 CPI가 캐쉬 스케일링의 기준이 되어야 한다[6]. 본 논문에서 제안된 캐쉬 스케일링 알고리즘은 사전에 정의된 기간마다 각 코어에서 실행된 태스크들의 CPI를 검사한다. 과거의 CPI와 현재 CPI 차의 절대값이 임계값(threshold)보다 크다면 캐쉬 스케일링 과정에 들어가게 된다. 캐쉬 스케일링 과정에서 현재 CPI가 과거 CPI보다 크다면 같은 기간 동안 실행된 명령어의수가 적었기 때문에 성능향상을 목적으로 캐쉬의 웨이를 늘린다. 반대의 경우 프로그램 실행에 적절한 캐쉬의 크기가 제공되고 있으므로 에너지 절감을 목적으로 캐쉬의 웨이를 줄인다. 가정하는 모델에서 온칩캐쉬는 모든 코어들이 공유하고 있기 때문에 캐쉬구조의 조절은 모든 코어들의 정보를 토대로 결정한다. 따라서, 워킹셋이 큰 태스크와 워킹셋이 작은 태스크가 동시에 실행된다면 일반적으로 워킹셋이 큰 태스크의 요구가 반영되어 캐쉬 크기가 결정될 것이다.

```
If ( abs( cur_cpi - last_cpi) > threshold)
   If ( cur_cpi > last_cpi )
      If ( cache way != MAX )
         Incr cache way;
   else
      if ( cache way != MIN )
         decr cache way;
 else
      nothing change;
```

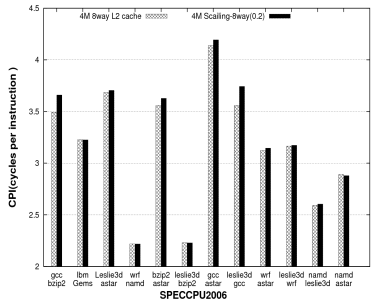
**논문에서 제안된 캐쉬 스케일링 알고리즘**

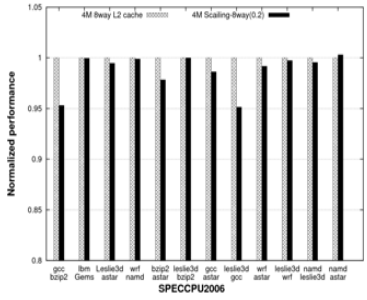
**4. 성능 평가** 본 논문에서 제안된 스케일링 기법과 시스템 디자인을 위해 풀시스템 시뮬레이터인 시믹스(Simics 3.0)[7]를 이용하였다. 시믹스에서 사용된 운영체제로는 리눅스 커널 2.6.28버전을 사용하였고 SPEC CPU2006을 이용하여 성능을 측정하였다. 태스크의 특성이 온칩 캐쉬에 미치는 영향을 부각시키기 위해 순차실행 프로세서 구조(in-order processor)를 이용하였고 넓은 범위의 스케일링을 제공하기 위해 8 웨이의 셋-어소시에티브 캐쉬를

기본 캐쉬 구조로 설정하였다. 캐쉬 스케일링 알고리즘은 10억 싸이클(cycle)에 한 번씩 실행된다. 시뮬레이션 결과를 이용하여 프로세서에서 소비된 전력을 측정하기 위해서 McPAT[8]을 이용하였다.

**실험 1. 임계값에 따른 실험 결과** 본 연구에서 제시한 알고리즘에서 적절한 임계값의 설정은 알고리즘에서 중요한 요소이다. 8개의 워크로드를 2개씩 조합하여 시뮬레이션한 결과 임계값이 0.5인 경우 평균 80회 스케일링 되었으며 1.2%의 성능 감소로 7.8%의 에너지를 절감할 수 있었다. 임계값이 0.2인 경우 평균 210회 스케일링 되었으며 1.4%의 성능감소로 8.4%의 에너지를 절감할 수 있었다. 실험 결과를 바탕으로 실행되는 프로그램에 따라 동적으로 임계값을 변화시킴으로써 더 좋은 결과를 얻을 수 있음을 예상할 수 있다.

**실험 2. 실험 결과** 실험은 SPEC CPU2006에서 8개의 워크로드를 선택하고 2개씩 조합하여 시뮬레이션하였다. 각 워크로드 세트는 평균 약 4,100억개의 명령어를 실행하였고 시뮬레이션 결과를 이용하여 성능과 에너지를 분석하였다. 3장에서 제안된 알고리즘의 임계값은 0.2로 설정하였다. 그림 2는 캐쉬 스케일링에 따른 성능의 감소를 보여준다. 실험에 사용된 워크로드에서 평균 1.3%의 성능 감소를 보였다. Leslie3d와 astar로 구성된 워크로드 세트를 제외한 CPI가 큰 워크로드 세트의 경우 높은 성능 감소를 보였는데 이것은 CPI가 큰 워크로드의 경우 CPI가 작은 워크로드에 비해 같은 임계값에 대해 민감하게 반응하는 것을 알 수 있다. CPI가 큰 워크로드는 많은 캐쉬 스케일링 기회로 인해 더 많은 에너지를 절감하는 것을 그림 3을 통해 확인할 수 있다. 논문에서 제안된 기법을 통해 평균 8.4%의 에너지를 절약할 수 있었다.



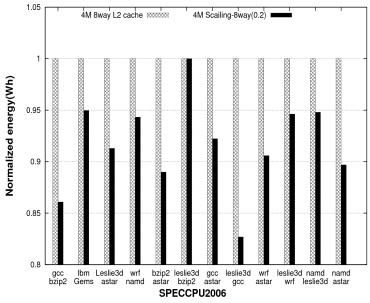




그림 1. 실험 워크로드 세트의 CPI. 그림 2. 실험 워크로드 세트의 성능. 그림 3. 캐쉬 스케일링에 따른 에너지

**5. 결론** 본 논문에서는 멀티 코어 환경에서의 온칩캐쉬 스케일링을 통해 프로세서 내에서 캐쉬에 의해 소비되는 에너지를 절감하는 기법을 제안하였다. 본 연구에서는 셋-어소시에티브 캐쉬에서 웨이의 수를 변화시킴으로써 캐쉬의 크기를 변화시켰고 제안된 알고리즘으로는 동적으로 변화하는 CPI 분석을 통해 캐쉬 스케일링을 결정하였다. 제안된 기법을 이용한 시뮬레이션 결과 평균 1.3%의 성능 감소로 8.4%의 에너지를 절약할 수 있음을 보였다. 향후 제안된 알고리즘을 토대로 두 가지 보완을 계획하고 있다. 먼저 실행되는 프로그램의 특성에 따라 동적으로 임계값이 변화되도록 수정되어야 한다. 두 번째로는 현재 실행되는 프로그램의 CPI에 따라 다른 임계값이 제공되도록 스케일링 알고리즘을 확장시킬 계획이다. 또한 여러 워크로드중에서 최소의 성능 감소로 최적의 에너지 절약을 할 수 있는 적절한 워크로드 페어링(pairing)을 제공하는 운영체제 스케쥴링 기법을 통해 스케일링 캐쉬의 효과를 더욱 증가시킬 계획이다.

**참고문헌**

[1] David H. Albonesi, "Selective Cache Ways : On-Demand Cache Resource Allocation," Proc. of 32 Intl. Sym. on *Microarchitecture*,vol.32,pp.248-259,1999.

[2] Yang.SH, Michael D. Powell, Babak Falsafi, Kaushik Roy, T.N. Vijaykumar, "An Integrated Circuit/Architecture Approach to Reducing Leakage in Deep-Submicron High-Performance I-Caches", Proc. of 7 Intl. Sym. on HPCA ,vol.7,pp.147,2001

[3] Yang.SH, Powell. MD, Falsafi. B, Vijaykumar. TN, "Exploiting choice in resizable cache design to optimize deep-submicron processor energy-delay", Proc. of 8 Intl. Sym.on *High-PerformanceComputerArchitecture*,vol.7,pp.151-161,2002

[4] Balasubramonian. R, Albonesi. D, Buyuktosunoglu. A, Dwarkadas. S, "Memory hierarchy reconfiguration for energy and performance in general-purpose processor architectures",Proc.of 33 Intl.Sym.on *Microarchitecture*,vol.33, pp.245-257,2000.

[5] Dhodapkar. AS, Smith. JE, "Managing multi-configuration hardware via dynamic working set analysis", *ACMSIGARCHComputerArchitectureNews*,vol30,no.2,pp.233-244,2002

[6] David Tam, Reza Azimi, Livio Soares, Michael Stumm, "Managing shared L2 Caches on Multicore Systems in Software", In Proc. of Workshop on the *InteractionbetweenOperatingSystemsandComputerArchitecture*,2007

[7] Magnusson. PS, Christensson. M, Eskilson. J, Forsgren. D, Hallberg. G, Hogberg. J, Larsson. F, Moestedt. A, Werner. B, "Simics: A full system simulation platform", *COMPUTER*,vol35,no.2,pp.50-58,2002

[8] Li.S, Ahn.JH, Strong.RD, Brockman.JB, Tullsen.DM, Jouppi.NP,"McPAT:an integrated power, area, and timing modeling framework for multicore and manycore architectures", Proc. of 42 Intl. Sym.on *Microarchitecture*,vol.42, pp.469-480, 2009.