Journal of Knowledge Information Technology and Systems

ISSN 1975-7700

http://www.kkits.or.kr

Adaptive Set-Division Replacement Policy for Increasing Cache Efficiency

Young-Il Cho[1*], **Sang-Jeong Lee**[2]

[1]*Department of Computer Science, University of Suwon*
[2]*Department of Computer Engineering, Soonchunhyang University*

ABSTRACT

Modern processors have large on-chip caches to mitigate off-chip memory latency. The Least Recently Used(LRU) replacement policy represents the cache blocks in a set as LRU stack. This policy picks the LRU block as the replacement candidate. In most condition, it can accomplish the work well, but for some memory-intensive workload it can not provide a long enough access history for a given cache size and associativity. Also in some workload, the majority of lines go through the cache space without making any sense. Cache performance can be improved if a long range of access history can be held so that some period of records can contribute to cache hits.

In this paper, we propose an Adaptive Set-division Replacement Policy(ASRP) for effective cache management. In ASRP policy, the set of the last-level cache is divided into several subsets, only one subset is activated when replacement happens and the replacement area is limited to this subset using LRU policy. The cache misses are counted when a certain subset is activated. When the miss-count exceeds a threshold, the next subset is activated. So the threshold can indirectly decide the range of access history through the last-level cache. We use a sample method to dynamically determine the threshold for different workloads and different run-time phases of a certain workload. The experiment results show that ASRP reduces the average MPKI of the baseline 1MB 16-way L3 cache by 4.6%.

KEYWORDS: cache replacement policy, cache hits, ASRP, LRU policy, access history

ARTICLE INFO:

[*]Corresponding author is with the Department of Computer Science, University of Suwon, 17 Wauangil Bongdam-Eup Hwaseong-Si, Gyeonggi-Do, 445-743, KOREA.
E-mail address: yicho@suwon.ac.kr

1. 서 론

CPU와 오프-칩 메모리간의 성능 차이 증가로 고성능 마지막-수준 캐시(Last-Level Cache : LLC)의 필요성이 한층 중요시되고 있고 잘 설계된 LLC는 CPU와 메모리간의 성능 차이를 메울 수 있는 가장 효과적인 방법이다. 최근 프로세서는 시스템 성능을 개선하기 위해 대용량의 LLC를 제공하지만 대부분 워크로드들의 워킹-셋 크기가 크기 때문에 속도가 느린 오프-칩 메모리 경과시간을 완전히 감추지는 못한다. 따라서 마이크로프로세서 설계 분야에서 캐시를 효율적으로 이용할 수 있는 최적의 캐시 교체정책이 중요한 연구 과제이다.

LRU(Least Recently Used) 교체정책은 과거 수십 년간 온-칩 프로세서에서 가장 일반적으로 사용된 캐시 교체 정책이다. LRU 정책은 유효한 캐시 액세스 히스토리를 유지할 수 있다. 따라서 높은 지역성을 갖는 워크로드에 대해 특히 좋은 성능을 얻을 수 있다. 그러나 가용 캐시 크기보다 큰 워킹-셋을 갖는 메모리 집중적인 워크로드에서 LRU 정책은 캐시 공간을 효과적으로 사용할 수 없고 성능도 유동적이다. 이런 문제점을 해결하기 위해 LRU 정책의 성능을 개선시키는 많은 연구들이 있었다. DIP(Dynamic Insertion Policy) 방법[1]은 메모리-집중적인 워크로드에서 캐시 성능을 개선시킨다. DIP는 교체를 희생자 선택 정책과 삽입 정책으로 나눈다. DIP의 삽입 정책은 캐시 적중에 기여한 블록들이 오랫동안 캐시 공간에 있도록 하기 위해 삽입할 블록들을 LRU 위치에 놓는다. MLP-인지 정책[2]은 미스 처리 단가가 높은 미스의 수를 감소시키기 위해 각 캐시 블록에 대한 MLP(Memory Level Parallelism)-기반 단가를 계산하는 실행시간 기술을 제시했다. 데드 라인 예측[3,4,5]은 주어진 라인의 마지막 터치를 예상한다. 다시 액세스되지 않으리라 간주되는 라인은 교체를 위한 높은 우순순위가 주어진다. [6]은 전역 교체 알고리즘을 제안했다. 이 알고리즘은 캐시 계층의 각 레벨에 대해 독립적으로 동작하는 지역 교체 알고리즘 대신에 전체 캐시 계층을 고려한 전역 교체 알고리즘이다. [7]은 새로 들어온 불럭을 계층구조의 어디에 배치해야 하는 지를 결정함에 의해 캐시 계층구조에서 블록들을 관리한다. [8,9,10]은 LRU 정책을 촉진시키는데 목표를 두었다. 단가(캐시 블록에 관련된 수치적 속성)를 고려한 변형 LRU 정책이다. LRU의 문제점을 개선시키기 위해 제안된 방법들은 대부분 큰 하드웨어 오버헤드를 요구하고 특정 워크로드에서만 잘 동작한다. LLC를 위한 바람직한 교체정책은 가까운 장래에 다시 참조되지 않을 블록들을 교체시켜 캐시가 유용한 블록만 갖도록 하므로 서 좋은 성능을 갖게 해야 한다.

본 연구에서 두 요인이 주로 캐시 교체 정책의 성능에 영향 줌을 확인하였다. 첫째, 최근의 캐시 액세스 히스토리에 대한 처리 방법이다. LRU 정책은 새 캐시라인에 더 높은 우선순위를 주어 그것이 오랫동안 캐시 공간에 머물도록 한다. 둘째, 충분한 액세스 히스토리를 저장할 수 있는 능력이다. 워킹 셋이 가용한 캐시 크기보다 클 때 긴 히스토리를 저장할 수 있으면 더 많은 캐시 라인을 적중시킬 수 있다. LRU 정책에서는 액세스 히스토리의 길이가 캐시 크기와 연

관(associativity)에 의해 제한되므로 충분한 액세스 히스토리를 갖는데 제약받는다.

본 논문에서는 적은 하드웨어 오버헤드로 여러 워크로드에서 잘 동작하는 캐시 관리 정책인 SRP(Set-division Replacement Policy)를 제안한다. SRP는 캐시 셋을 여러 서브셋으로 분할하고 교체는 한 서브셋(활성 서브셋)에서만 일어나고 나머지 서브셋들(비활성 서브셋들)은 어떤 변화도 발생하지 않으며, 교체가 일어나는 서브셋에서는 LRU 정책을 사용하는 새로운 교체정책이다. 어떤 서브셋이 활성화되면 캐시 미스가 카운트되며 그 카운트 값이 임계값을 초과하면 다음 서브셋이 활성화된다. 따라서 임계값은 임의 서브셋이 가질 수 있는 액세스 히스토리 길이에 영향 줄 수 있다. 결국, 서브셋들로의 분할은 LRU 교체정책보다 더 긴 액세스 히스토리를 갖게 할 수 있다.

워크로드에 적합한 액세스 히스토리 길이는 응용마다 다를 수 있고 또한 임의 응용의 실행시간 중에도 변할 수 있다. 본 논문에서는 여러 응용뿐 아니라 한 응용의 여러 실행시간 단계에 대해 임계값을 동적으로 결정하는 적응적 방법인 ASRP(Adaptive Set-division Replacement Policy)을 제안한다. 실험 결과 간단한 변경으로 메모리 집중적인 워크로드들에서 캐시 성능을 상당히 개선시고 LRU 친화적인 워크로드들에서도 개선됨을 보여준다.

2. 이론적 배경

캐시 계층의 마지막-수준 캐시에서의 미스는 프로세서를 오랜 동안 멈추게 한다. 캐시 교체정책은 장래 액세스에서 최대의 적중을 얻을 수 있도록 하기 위해 셋에 유용한 블록들을 유지해야한다.

캐시로의 블록 액세스 패턴들을 관찰하면 각 블록을 4개 카테고리로 분류할 수 있다. 첫째, 블록이 빈번히 액세스되고 연속적 액세스간의 간격이 작은 경우로 이들 블록은 높은 시간적 지역성을 갖는다. 둘째, 블록이 짧은 기간 동안 빈번히 액세스되고, 어떤 기간 동안은 액세스되지 않은 다음, 다시 짧은 기간 동안 빈번히 액세스되는 경우로 주기적으로 높은 지역성을 갖는다. 셋째, 블록이 일관되게 액세스되나 연속적 액세스간의 간격이 첫 번째 카테고리보다 긴 경우이다. 넷째, 블록이 한 번 액세스되고 다시 액세스되지 않거나 한 참 후에 다시 액세스되는 경우이다.

첫 번째와 두 번째 카테고리는 짧은 시간 간격을 두고 계속 액세스되므로 캐시에 저장하면 적중률을 높이게 되며 블록을 빨리 CPU에 양도하게 되어 워크로드의 실행시간을 개선시킬 수 있다. 그러나 두 번째 카테고리의 경우 캐시 기록 히스토리가 짧은 경우 재 액세스되기 전에 퇴출될 수 있으므로 미스를 유발할 수 있다. 세 번째 카테고리도 액세스 히스토리가 짧은 경우 재 액세스되기 전에 퇴출되므로 미스를 유발할 수 있다. 네 번째 카테고리는 자주 액세스되는 블록이 아니므로 많이 액세스되는 불록을 위해 이들 카테고리는 빨리 퇴출시키는 것이 유용하다.

기존의 LRU 교체정책은 여러 응용의 다양한 요구를 만족시킬 수 없다. LRU 교체정책은 몇 가지 문제점을 갖는다. 첫째, 캐시 액세스의 최신 정보만 고려한다. 즉, LRU는 가장 최근에 사용된 블록이 곧 다시

사용할 확률이 가장 높다고 가정한다. 따라서 메모리로부터 새로 가져온 블록이나 캐시에서 적중된 블록을 MRU(Most Recently Used) 위치로 이동시킨다. 이는 자주 사용되지 않는 블록에 의해 캐시가 오염되고 자주 사용되는 블록이 LRU 스택으로부터 퇴출될 수 있는 문제점을 갖는다. 둘 째, 워킹-셋이 가용 캐시보다 클 때 반복적인 액세스 패턴을 갖는 경우 캐시 스래싱 문제가 발생한다. 즉, LRU는 가장 최근에 사용된 블록들을 캐시에 유지하기 때문에 반복적인 액세스 패턴의 경우 블록이 재사용되기 전에 캐시로부터 퇴출되어 대량의 미스가 발생하게 된다. 워킹-셋이 가용 캐시보다 큰 응용들은 긴 캐시 액세스 히스토리를 요구한다. 마지막으로 워크로드는 실행시간동안 워킹 셋이 변할 수 있다. 그러나 LRU는 그런 실행시간 요구를 만족할 시킬 수 없다.

LRU를 변형한 교체 알고리즘들은 삽입 및 프로모션 정책을 변경하여 개선시킬 수 있다고 하였다. 원본 LRU 정책은 액세스된 블록들을 MRU 위치로 프로모션 시키고, 들어오는 블록들은 MRU 위치로 삽입한다. Quresh는 종종 들어오는 블록을 LRU 위치에 놓는 것이 유익하다고 하였다[1]. Liu는 블록을 스택의 위쪽으로 그러나 MRU에서 멀지 않는 곳에 삽입하는 것이 이득을 준다고 하였다[4].

본 연구는 기존의 LRU 교체정책을 개선시키고 마지막-수준 캐시 미스를 감소시키는데 초점을 둔다. 캐시 공간을 효율적으로 이용하기 위해 캐시 셋을 여러 서브셋으로 분할하고 응용들에 적합한 액세스 히스토리 길이에 대한 요구를 적응적으로 만족시키는 것을 목표로 한다.

3. 실험방법

3.1 베이스라인 구성

본 논문에서 모든 실험을 위해 트레이스 구동 캐시 시뮬레이터를 사용한다[11]. 표1은 베이스라인 캐시 구성을 보여준다. 베이스라인 L1 명령어 캐시는 32KB 2-way 셋 연관(set associative)이고 데이터 캐시는 4-way 셋 연관이다. 베이스라인 L2 캐시는 256MB 8-way 셋 연관이다. L3 캐시는 1MB 16-way 셋 연관이고 베이스라인의 모든 레벨의 캐시는 64B 라인 크기를 사용한다. L3 캐시 액세스 트레이스 파일들을 생성하기 위해 Pin[12]을 사용한다. 표에서 LRU는 기존의 LRU 교체 정책을 의미하고 LRU 정책을 베이스라인으로 사용한다.

표1. 베이스라인 캐시 구성
Table 1. Configuration of baseline cache

L1 Inst/Data	32KB, 64B 라인, 2-way/4-way, LRU, 1 cycle latency
L2	256KB, 64B 라인, 8-way, LRU, 8 cycle latency
L3	1MB, 64B 라인, 16-way, LRU, 20 cycle latency

3.2 벤치마크

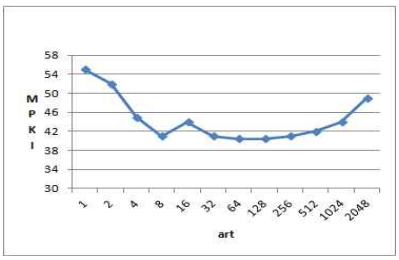
실험에서 SPEC CPU2000 벤치마크를 사용한다. 콜드 스타트(cold start) 미스의 영향을 제거하기 위해 준비시간(warm-up)을 갖고 reference 입력을 사용하여 SimPoint로 부터 각 벤치마크에 대해 250M 명령어를 얻었다. 교체정책은 강제 미스(compulsory

miss : 블록을 처음 액세스할 때 캐시에서 발생하는 미스)에는 영향 주지 않으므로 강제 미스가 캐시미스의 큰 백분율을 차지할 경우 교체정책은 캐시 성능에 영향주지 못한다. 따라서 26개 벤치마크로부터 강제 미스가 전체 캐시미스의 50% 미만인 16개 워크로드들을 실험에서 선택했다.

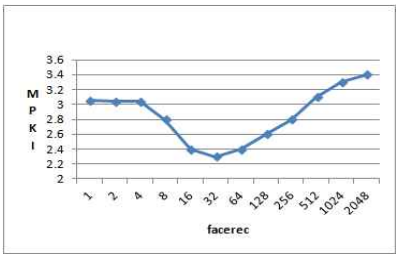
4. 셋-분할 교체정책(Set-division Replacement Policy : SRP)

기존의 캐시 관리 정책은 액세스 히스토리를 유지하기 위해 대표적으로 LRU 교체정책을 사용한다. LRU 교체정책은 최근에 액세스된 캐시 라인들이 캐시 적중에 기여할 수 있도록 오랜 시간동안 캐시에 간직한다. 그러나 워크로드가 가용 캐시 크기보다 큰 워킹 셋을 가질 때 LRU 교체정책은 충분히 긴 액세스 히스토리를 유지할 수 없어 대부분의 액세스된 캐시 라인들은 캐시 적중에 기여하지 못하면서 캐시 공간을 차지하게 된다.

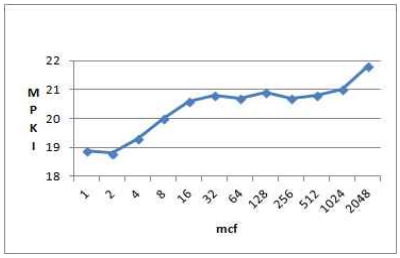
본 논문에서는 캐시(L3) 셋을 4개 서브셋으로 분할하고, 임의 시간에는 단지 한 서브셋만 활성화되며, 교체는 현 활성화 서브셋에서만 발생하고 활성화 서브셋에서는 LRU 교체 정책을 사용하는 캐시 교체정책을 제안한다. 즉, 캐시 미스가 발생하면 현 활성 서브셋의 LRU 블록이 퇴출자로 선택되고, 캐시 적중 시에 들어오는 블록은 활성에 상관없이 해당 블록에 대응하는 서브셋의 MRU 위치로 이동시킨다. 캐시 미스들은 임의 서브셋이 활성화될 때 카운트된다. 미스-카운트가 임계값을 초과하면 다음 서브셋이 활성화된다. 따라서 어떤 서브셋이 가질 수 있는 액세스 히스토리 길이를 임계값을 통해 간접적으로 결정할 수 있다. 임계값이 큰 수로 설정되면 오랜 기간 동안 활성 서브셋에서만 교체가 발생하므로 비활성 서브셋들은 오래된 캐시라인들을 유지할 수 있다. 따라서 임계값이 큰 경우 4개 서브셋으로 분할하는 것은 더 긴 액세스 히스토리를 가질 수 있기 때문에 메모리-집중적인 워크로드에서 장점을 갖는다. 임계값이 작은 수로 설정되면 활성 서브셋이 빠르게 변경되고 이 경우 교체 영역은 짧은 시간에 전체 캐시 공간을 커버하게 된다. 따라서 기존의 LRU 교체 정책같이 동작하므로 LRU 친화적인 워크로드에 대해서도 장점을 갖는다. 각 워크로드에 대해 임계값을 어떻게 설정할 것이냐는 캐시 성능을 개선시키는데 중요 역할을 한다.



(a) art



(b) facerec



(c) mcf

그림1. 벤치마크 프로그램에서 임계값의 변화에 따른 MPKI

Figure 1. MPKI for different thresholds in benchmark programs

그림1은 각 워크로드에 대한 임계값의 영향을 보여준다. 임계값을 1에서 2K로 변화시키면서 MPKI를 측정하였다. 임계값이 128을 초과하면 변화가 미소하거나 오히려 미스 율이 증가하는 결과를 보인다. 또한 대부분의 워크로드에 대해 적합한 임계값이 있다는 것을 확인할 수 있다. art는 큰 임계값에서 성능이 개선시키는데 이는 긴 액세스 히스토리를 갖는 것이 이런 종류의 프로그램에서 이점을 갖는다는 것을 의미한다. facerec은 중간 임계값에서 좋은 결과를 얻는다. mcf 같은 LRU 친화적인 워크로드는 작은 임계값에서 좋은 결과를 얻는다. 결론적으로 가장 큰 임계값을 128로 취하는 것이 대부분의 워크로드에서 잘 동작하는 것을 확인하였다. 따라서 하드웨어 구현을 고려하여 워크로드들에 적합한 임계값을 실행시간에 찾기 위해 8개 임계값(1, 2, 4, 8, 16, 32, 64, 128)을 사용한다.

SRP의 하드웨어 구조는 그림2와 같다. 16-ways L3의 각 셋을 4개 서브셋으로 분할한다. 셋에서 2비트의 활성-서브셋은 현재 활성화된 서브셋을 지시하고, 7비트의 셋-미스 카운터는 현 활성 서브셋의 미스 수를 카운트하는데 사용되며 그 값이 임계값 레지스터에 있는 값보다 클 때 활성 서브셋은 이웃 서브셋으로 변경된다. 이때 활성-서브셋과 셋-미스 카운터는 0으로 초기화되고 임계값 레지스터의 값은 고정 값(2의 지수값)으로 초기화한다.

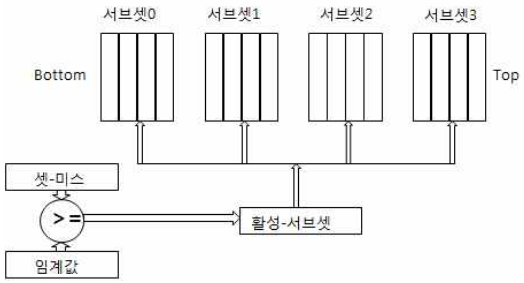


그림2. SRP의 하드웨어 구조

Figure 2. Hardware of SRP

5. 적응적 셋-분할 교체정책(Adaptive Set-division Replacement Policy : ASRP)

이 장에서는 실행시간에 적합한 임계값을 동적으로 선택하는 ASRP의 구조와 하드웨어 오버헤드에 대해 알아본다.

5.1 ASRP 구조

그림1에서와 같이 각 워크로드는 자신에 적합한 임계값을 갖는다. 또한 어떤 워크로드는 여러 실행-시간 단계마다 다른 임계값을 요구할 수도 있다. 따라서 여러 워크로드와 실행시간의 여러 단계에 적합한 임계값을 동적으로 선택하는 메커니즘이 필요하다. 본 논문에서는 적절한 임계값을 동적으로 선택 위해 표본 방법인 Set Dueling[13]을 사용한다. 캐시 셋들을 표본 셋(sample sets)과 추종 셋(follow sets)으로 구별한다.

전체 캐시에 평균적으로 분산되어있는 표본 셋은 고정 임계값을 갖는다. 표본 셋에서 발생된 미스들은 카운트된다. 추종 셋은 어떤 임계값을 선택할지를 동적으로 결정하기 위해 표본 셋들의 카운터들을 사용한다. 본 논문에서, 8개 임계값 각각에 대해 4개 표본 셋을 할당한다. 32 표본 셋은 캐시 셋들에 고르게 분포된다. 나머지 셋들은 추종 셋이다. 그림3은 ASRP의 구조를 보여준다. 각 셋에 4비트가 추가된다. 첫 번째 비트는 해당 셋이 표본 셋인지 추종 셋인지를 나타내고 나머지 3비트는 8개 임계값 중 어떤 임계값에 할당되어있는지를 나타낸다. 8개의 카운터는 각 표본 셋에서 미스 수를 카운트하기 위해 사용된다. 캐시 미스가 특정 표본 셋에서 발생하면 대응 카운터가 1 증가한다. 교체가 표본 셋에서 발생할 때 임계값은 대응하는 고정 값으로 설정하고 미스 카운트가 그 임계값에 더해진다. 교체가 추종 셋에서 발생할 때 더 적은 미스를 갖는 임계값을 선택하기 위해 8개 중 가장 작은 값을 갖는 카운터에 대응하는 임계값이 선택된다.

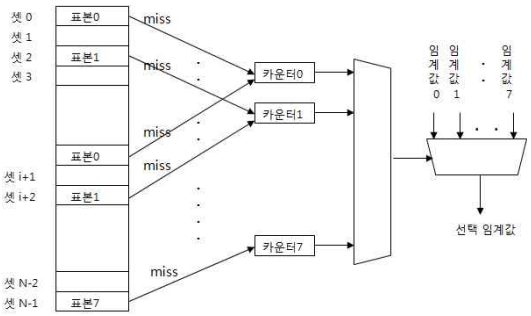


그림3. ASRP의 하드웨어 구조

Figure 3. Hardware of ASRP

그림4는 8개 임계값과 ASRP에 대한 정규화된 MPKI를 보여준다. 대부분의 워크로드에서 ASRP는 적절한 임계값을 찾을 수 있고, 워크로드 art의 경우 ASRP는 최고의 정적 선택보다 더 좋은데 이것은 ASRP가 워크로드의 여러 실행시간 단계에 적합한 임계값을 동적으로 선택하기 때문이다. 또한 워크로드 ammp의 경우 ASRP는 최상의 정적 성능을 유지하는데 실패하지만 크게 차이나지는 않는다. art에 대해 ASRP는 최대 32% MPKI를 감소시킨다. 모든 벤치마크에 대해 ASRP는 평균 4.6% 만큼 MPKI를 감소시킨다.

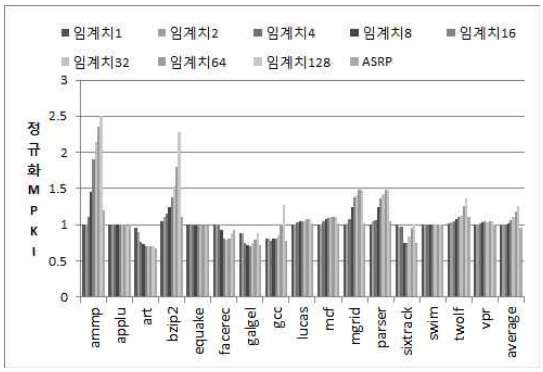


그림4. ASRP의 정규화된 MPKI

Figure 4. Normalized MPKI of ASRP policy

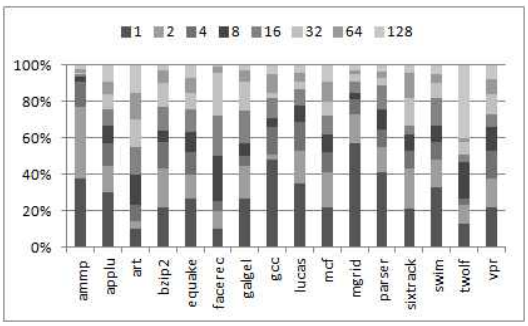


그림5. ASRP 정책에서 추종 셋에 의해 선택된 임계값 백분율

Figure 5. Percentage of threshold selected by following

sets in ARP policy.

그림5는 ASRP 정책에서 추종 셋에 의해 선택된 임계값의 백분율을 보여준다. 각 워크로드는 실행-시간에 적합한 임계값이 변한다. 각 워크로드에 대해 가장 큰 부분을 차지하는 임계값들은 대부분 정적으로 선택한 최상의 임계값과 같음을 확인하였다.

5.2 ARP의 하드웨어 오버헤드

ASRP 정책을 위한 하드웨어 오버헤드는 표2와 같다. 이는 원본 LRU 정책의 70%에 해당한다. 16-way LRU 캐시에서 들어오는 블록이 MRU 위치로 삽입될 때 모두 16 블록의 상태가 갱신돼야 하지만 제안 구조는 최대 4 블록의 상태만 갱신하면 되기 때문에 교체정책의 실행시간도 상당히 감소시킬 수 있다.

표2. ASRP의 하드웨어 오버헤드
Table 2. Hardware Overhead of ASRP

요소	오버헤드
카운터	32비트 * 8개
활성-서브셋	2비트 * 1024 셋
셋-미스 카운터	7비트 * 1024 셋
임계값 레지스터	7비트
표본/추종 플래그	4비트 * 1024 셋
LRU 스택	2비트 * 16K 라인
합계	45K 비트

6. 실험 결과

이 장에서는 시뮬레이션을 통해 제안한 교체정책인 ASRP의 여러 하드웨어 요소가 캐시미스에 주는 영향에 대해 분석한다. 즉, 캐시 크기, 서브셋 수, 임계값에 대한 표본 수를 변화시켰을 때 캐시미스에 주는 영향을 알아본다.

6.1 캐시 크기

ASRP의 캐시 크기에 대한 영향을 알아본다. 캐시 크기를 1MB~8MB로 변화를 주고 연관은 16-way를 유지한다. 그림6은 4개 캐시 크기(1, 2, 4, 8MB)에 대한 ASRP의 MPKI를 보여준다. MPKI는 베이스라인 1MB 16-way LRU 교체정책을 사용한 캐시에 대한 정규화된 수치이다. 즉, 1MB LRU 교체정책을 1로 했을 때 각 캐시 크기에 대한 정규화된 미스 율을 보여준다.

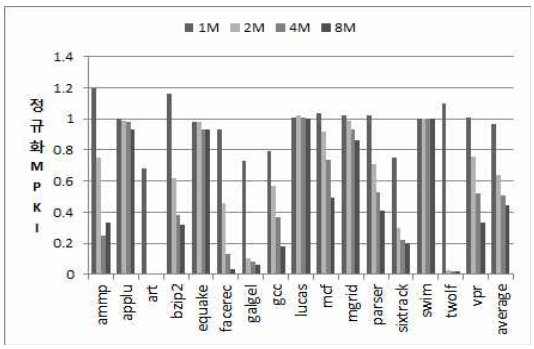


그림6. 여러 캐시 크기에 대한 ASRP의 정규화된 MPKI
Figure 6. Normalized MPKI of ASRP policy for different cache sizes.

ASRP는 art, galgel, gcc에 대해 MPKI를 크게 감소시킨다. 특히, twolf의 경우 캐시 크기가 1M일 경우 MPKI가 베이스라인보다 오히려 증가하였으나 2M이상일 경우 MPKI가 급격히 감소함을 보였다. 평균적으로 보면 캐시 크기가 2M 이상으로 커지면 가용

캐시 공간이 충분하므로 LRU와 ASRP의 MPKI가 비슷한 것을 확인하였다.

6.2 서브셋 수

서브셋 수는 중요한 의미를 갖는다. 너무 작게 설정되면 ASRP는 충분히 긴 액세스 히스토리를 저장할 수 없다. 한편, 너무 크면 많은 하드웨어 비용과 LRU에 친화적인 워크로드의 성능에 악영향을 준다.

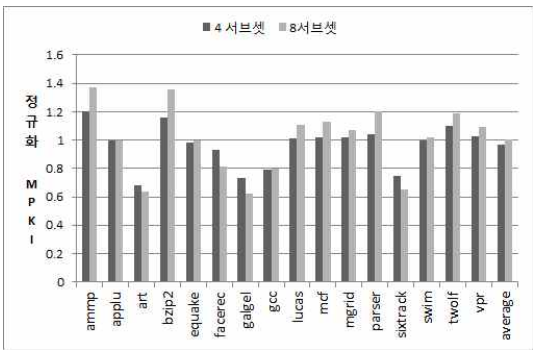


그림7. 여러 서브셋 수에 대한 ASRP의 정규화된 MPKI

Figure 7. Normalized MPKI of ASRP policy for different number of subsets.

그림7은 4 서브셋과 8 서브셋에 대해 LRU 정책에 정규화된 MPKI를 보여준다. 서브셋이 증가하면 액세스 히스토리가 효과적으로 길게 되므로 art, facerec, galgel, sixtrack 같은 큰 워킹 셋을 갖는 워크로드들에게는 이점이 된다. 그러나 LRU에 친화적인 워크로드의 경우 많은 서브셋은 전체 캐시 공간을 커버하는데 긴 시간이 걸리고 최근에 액세스된 캐시 라인들이 교체될 가능성이 높으므로 ammp, bzip2, mcf, parser와 같이 캐시미스가 증가한다. 평균적으로 보면 8 서브셋과 4 서브셋이 비슷한 MPKI를 보였다. 본 논문에서는 하드웨어 오버헤드를 고려하여 4개 서브셋을 실험결과를 구할 때 사용하였다.

6.3 각 임계값을 위한 표본 셋의 수

ASRP에서 임계값을 결정하기 위해 표본 방법을 사용한다. 적절한 임계값을 찾기 위해 적은 표본 셋을 사용한다. 그림8은 캐시 크기와 연관을 동일하게 유지하면서 표본 수 4, 8, 16에 대해 정규화된 MPKI를 보여준다.

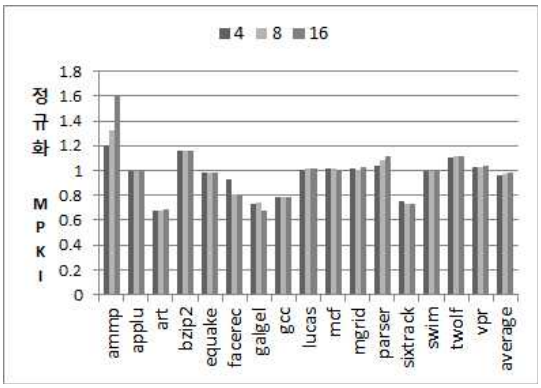


그림8. 여러 표본 셋 수에 대한 ASRP의 정규화된 MPKI

Figure 8. Normalized MPKI of ASRP policy for different number of sample sets.

대부분의 워크로드에 대해 표본 셋의 증가는 MPKI의 증가를 이끄는데 이것은 임계값을 취할 추종 셋의 수가 상대적으로 적기 때문이다. facerec, galgel 같은 워크로드는 표본 셋의 증가로 이득을 얻는다. 이는 증가된 표본 셋이 ASRP가 여러 실행시간 단계에 보다 적합한 임계값을 취하도록 만들

기 때문이다. 모든 워크로드를 고려했을 때 실험에서 각 임계값에 대해 4 표본-셋을 선택했다.

7. 결론

일반적으로 사용되는 LRU 교체정책은 메모리-집중적인 워크로드에서 빈약하게 수행한다. 이는 긴 액세스 히스토리를 가질 수 없기 때문이다. 본 논문에서 제안한 적응적 셋-분할 교체정책은 캐시를 4개 서브셋으로 분할하고 교체가 발생할 때 단지 한 서브셋만 활성화되고 교체 영역은 LRU 정책을 사용하는 활성화된 서브셋으로 제한되고 나머지 비활성 서브셋은 액세스 히스토리를 오랫동안 유지하므로 서 긴 액세스 히스토리를 요구하는 메모리 집중적인 워크로드에서도 적중률을 개선시킨다. 또한 여러 응용과 어떤 워크로드의 여러 실행시간 단계에 적합한 임계값을 동적으로 선택하는 간단한 방법을 제안하였다.

실험결과는 ASRP가 LRU 교체정책을 사용하는 베이스라인 1MB 16-way L3 캐시의 평균 MPKI를 4.6% 감소시키는 것을 보여주었다. 또한 ASRP는 1MB 16-way 셋 연관 LLC의 경우 LRU의 70% 하드웨어 오버헤드를 갖는다.

References

[1] M. K. Qureshi, A. Jaleel, Y. N. Patt, S. C. Steely Jr. and J. Emer, *Adaptive insertion policies for high-performance caching.* In ISCA'07, pp. 381-391, 2007.

[2] M. K. Qureshi, D. N. Lynch, O. Mutlu, and Y. N. Patt, *A case for MLP-aware cache replacement.* ISCA'06, pp. 167-178, 2006.

[3] M. Kharbutli and Y. Solihin, *Counter-Based Cache Replacement and Bypassing Algorithms.* IEEE Transactions On Computers, Vol.57, Issue 4, pp.433–447, 2008.

[4] H. Liu, M. Ferdman, J. Huh, and D. Burger. *Cache Bursts: A New Approach for Eliminating Dead Blocks and Increasing Cache Efficiency,* MICRO'08, pp. 222–233, 2008.

[5] S. M. Khan, Z. Wang and D. A. Jim´enez, *Decoupled Dynamic Cache Segmentation,* HPCA'12, pp. 1-12, 2012

[6] M. Zahran, *Cache Replacement Policy Revisited*, WDDD(Workshop on Duplicating, Deconstructing, and Debunking), 2007.

[7] M. Zahran and S. A. McKee, *Adaptive Block Placement Policy for Cache Hierarchies*, in SMART'09(Statistical and Machine learning approaches to ARchitectures and compilaTion), 2009.

[8] E. Perelman, G. Hamerlyherwood and B. Calder, *Using simpoint for accurate and efficient simulation.* SIGMETRICS'03, Vol. 31, Issue 1, pp. 318–319, 2003.

[9] J. Jeong and M. Dubois. *Optimal replacements in caches with two miss costs.* SPAA'99(Proceedings of the 11th Annual ACM Symposium on Parallel Algorithms and Architectures), 1999.

[10] D. A. Jiménez, *Insertion and Promotion for Tree-Based PseudoLRU Last-Level Caches*, MICRO'13, pp. 284-296, 2013

[11] SimPoint home page : http://cseweb.ucsd.edu/~calder/simpoint/

[12] Pin Homepage :

http://software.intel.com/en-us/articles/intel-software-development-emulator.

[13] M. Qureshi, A. Jaleel, Y. N. Patt, S. C. Steely and J. Emer, *Set-Dueling-Controlled Adaptive Insertion for High Performance Caching*, IEEE Micro Vol. 28, Issue 1, pp. 91-98, 2008.

캐시 효율성 향상을 위한 적응적 셋-분할 교체정책

조영일[1], 이상정[2]

[1] *수원대학교 컴퓨터학과*

[2] *순천향대학교 컴퓨터공학과*

요 약

최신 프로세서들은 오프-칩 메모리 latency를 완화하기 위해 온-칩 캐시를 갖는다. LRU 교체정책은 셋에 있는 캐시 블록들을 LRU 스택으로 나타낸다. 이 정책은 LRU 블록을 교체 대상자로 선택한다. 대부분의 경우에 잘 수행되나 일부 메모리 집중 워크로드에 대해 주어진 캐시 크기와 연관에 대해 충분한 액세스 히스토리를 제공하지 못한다. 또한 어떤 워크로드들에서는 대부분의 라인이 유용하게 사용되지 못하고 캐시 공간을 낭비한다. 긴 액세스 히스토리를 저장한다면 캐시 적중에 기여할 수 있고 따라서 캐시 성능을 향상시킬 수 있다.

본 논문에서는 효과적인 캐시 관리를 위해 적응적 셋-분할 교체정책(Adaptive Set-division Replacement Policy : ASRP)을 제안한다. ASRP 정책은 마지막-수준 캐시의 셋을 여러 서브셋으로 분할하고 교체가 발생할 때 한 서브셋만 활성화되며 LRU 정책을 사용하는 현재 활성 서브셋으로 교체 영역이 제한된다. 어떤 서브셋이 활성화될 때 캐시 미스가 카운트되며 미스 카운트가 임계값을 초과하면 다음 서브셋이 활성화된다. 따라서 임계값은 마지막-수준 캐시의 액세스 히스토리를 간접적으로 결정할 수 있다.

워크로드에 적합한 액세스 히스토리 길이는 응용마다 다를 수 있고 또한 임의 응용의 실행시간 중에도 변화할 수 있다. 여러 워크로드와 한 워크로드의 여러 실행-시간 단계에 대해 임계값을 동적으로 결정하는 간단한 방법을 제안한다. 실험결과 제안한 ASRP는 베이스라인 1 MB 16-way L3 캐시의 평균 MPKI를 4.6%를 감소시켰다.

Young-Il Cho received the BS, MS, PhD in electronic engineering from the Hanyang University in 1980, 1982 and 1985 respectively. He has been a professor in the Department of Computer Science at University of Suwon since 1986. His current research interests include computer architecture, high performance microarchitecture and wireless sensor network.

E-mail address: yicho@suwon.ac.kr

Sang-Jeong Lee received the BS, MS, PhD in electronic engineering from the Hanyang University in 1983, 1985 and 1988, respectively He is currently a professor at the department of computer science and engineering in Soonchunhyang University, Korea. His current research areas are computer architecture, network application and embedded system.

E-mail : sjlee@sch.ac.kr