로케이션 캐쉬 시스템의 효율을 개선한 스피시픽-웨이 캐쉬 시스템

윤상호° 이인환

한양대학교 전자컴퓨터통신공학과

shyun@csl.hanyang.ac.kr ihlee@hanyang.ac.kr

Specific-Way Cache System: An Efficient Location Cache System

Sangho Yun° Inhwan Lee

Dept. of Electronics and Computer Engineering, Hanyang University

요 약

집합-연관 캐쉬는 직접-사상 캐쉬보다 적중률이 높다는 장점이 있는 반면, 전력 소모가 많다는 단점이 있다. 그러한 단점을 보완하기 위해 웨이-프리딕팅 셋-어소시에이티브 캐쉬, 로케이션 캐쉬 시스템 등의 연구들이 계속 되어왔다. 본 논문에서는 로케이션 캐쉬 시스템에서 생각할 수 있는 논점들을 살펴보고, 이를 효율적으로 극복할 수 있는 스피시픽-웨이 캐쉬 시스템을 제안하였다. 또한 Simplescalar와 MiBench를 이용하여 스피시픽-웨이 캐쉬 시스템의 성능을 측정하였고, 그 결과 39.6%의 예상-적중률이 나타난 것으로 확인되었다.

1. 서 론

캐쉬(cache) 구성 시, 널리 쓰이는 두 가지 방법은 직접-사상 캐쉬(direct-mapped cache)와 집합-연관 캐쉬(set-associative cache)이다. 직접-사상 캐쉬는 구현이 용이하지만 실패(miss)가 많이 생긴다는 단점이 있는 반면, 집합-연관 캐쉬는 비교적 실패가 적은 대신 전력 소모에 있어 취약하다는 단점이 존재한다 [1]. 그러한 집합-연관 캐쉬의 단점을 보완하고자 많은 연구가 있었다 [2-7].

본 논문에서는 로케이션 캐쉬 시스템에서 발생하는 L2 캐쉬의 태그 및 비교기 중복, 그리고 사용되지 않은 채 폐기되는 로케이션 캐쉬 [6] 의 접근 방식을 개선하여, 보다 효율적인 스피시픽-웨이 캐쉬를 제안하였다.

2. 관련 연구

2.1. 페이즈드 캐쉬

전력 소모 문제를 해결하기 위해 Atsushi Hasegawa 등은 페이즈드 캐쉬(Phased Cache)를 제안하였다 [3]. 페이즈드 캐쉬는 집합-연관 캐쉬에서의 접근(access)을 두 단계로 나누었다. 첫 번째, 각 웨이의 태그 열(tag arrays)이 병렬 접근되고, 이 과정 동안 데이터 열은 접근되지 않는다. 두 번째, 태그 열의 비교 후 적중 되면 데이터 열(data arrays)이 접근 된다. 이렇게 페이즈드 캐쉬는 전력소모를 감소시키기 위해 태그 열과 데이터 열을 분리시켜 접근함으로써, 실패시의 불필요한 데이터 열 접근을 억제한다.

2.2. 웨이-프리딕팅 셋-어소시에이티브 캐쉬

웨이-프리딕팅 셋-어소시에이티브 캐쉬(Way-Predicting Set-Associative Cache)는 집합-연관 캐쉬에서 각 웨이가 병렬로 접근되기 전에 한 개의 웨이만을 예상(predict)하여 접근하는 기술이다 [5]. 예상이 적중(prediction-hit)되면, 예상했던 웨이의 캐쉬 접근은 성공적으로 완료되고 해당 웨이의 전력 소모만이 일어난다. 그러나 예상이 실패(prediction-miss)하게 되면 캐쉬는 그 후 남아있는 다른 웨이를 검색하고, 그로 인해 캐쉬 접근 시간이 증가한다. 즉 적중률이 높을 수록 웨이-프리딕팅 셋-어소시에이티브 캐쉬의 전력 소모가 줄어들 수 있다.

2.3. 로케이션 캐쉬 시스템

여러 웨이 가운데 앞으로 참조할 웨이에 대한 정보를 로케이션(location)이라 한다면, 그 정보가 저장되는 캐쉬를 로케이션 캐쉬(Location Cache)라 하고, 로케이션 캐쉬가 내장되어 있는 시스템을 로케이션 캐쉬 시스템(Location Cache System)이라 한다 [6].

로케이션 캐쉬는 L1 캐쉬와 병렬로 접근된다. L1 캐쉬에서 실패가 일어나고 이어서 로케이션 캐쉬에서 적중이 일어나게 되면, L2 캐쉬는 로케이션 캐쉬에 의해 출력된 웨이 정보(way information)를 입력 받는다. 로케이션 캐쉬로부터 입력 받는 웨이 정보가 올바르다면, L2 캐쉬는 그 정보에 따라 직접-사상 캐쉬처럼 동작된다. 물론 L2 캐쉬가 직접-사상 캐쉬처럼 동작했다 하더라도, 특정 웨이로 접근된 L2 캐쉬에서 실패가 생길 수 있다. 로케이션 캐쉬에서 출력된 웨이 정보가 올바르지 않을 수도 있기 때문이다 [6].

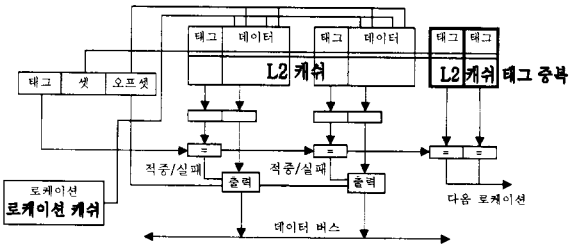

그림 1 로케이션 캐쉬 시스템의 블록다이어그램 [6]

3. 로케이션 캐쉬 시스템 분석

로케이션 캐쉬 시스템에는 생각해 볼 만한 여러 논점이 존재한다. 본 논문에서는 그 중 두 가지 사항에 관해 다루어 보고자 한다.

로케이션 캐쉬 시스템은 컨벤셔널 캐쉬 시스템(conventional cache system)에 비하여 추가의 태그 열 공간을 요구한다. 그림 1 은 Rui Min 등이 제안했던 로케이션 캐쉬 시스템이다. 이는 L2 캐쉬의 연관 정도가 2일 때의 구성을 도식화 한 것으로, 컨벤셔널 캐쉬 시스템에 로케이션 캐쉬가 추가되어있다. 문제는 L2 캐쉬의 기본적인 태그 열 외에, L2 캐쉬의 연관 정도 만큼의 태그 열이 더 중복되어야 한다는 데에 있다 [6]. 가령 L2 캐쉬의 연관 정도가 8이라면, 총 16개의 태그 열이 필요하다. 로케이션 캐쉬 시스템에서는 중복된 L2 캐쉬 태그를 통해 웨이 정보를 만들어내도록 설계되어 있기 때문에 이와 같이 추가의 태그 열이 필요하다. 또한 이렇게 추가의 L2 캐쉬 태그 열이 필요하게 됨에 따라, 각 태그 열을 위한 비교기도 L2 캐쉬의 연관 크기만큼 더 요구된다.

이에 덧붙여 로케이션 캐쉬 시스템에서 생각해 볼 만한 사항이 하나 더 존재한다. 바로, 로케이션 캐쉬가 항상 L1 캐쉬와 병렬적으로 동작한다는 점이다 [6]. 그림 2 에서 로케이션 캐쉬는 L1 캐쉬와 같은 레벨 상에 존재한다. 즉 L1 캐쉬를 접근할 때마다 항상 로케이션 캐쉬가 접근된다는 뜻이다. 그림 3 의 로케이션 캐쉬 시스템 알고리즘에서, 로케이션 캐쉬로부터 출력되는 웨이 정보가 사용되어지기 위해서는, L1 캐쉬가 반드시 실패되어야 한다는 것을 알 수 있다. 그러나, 문제는 L1 캐쉬가 적중할 때마다, 로케이션 캐쉬가 출력하는 웨이 정보는 매번 폐기 된다는 데에 있다 [6].

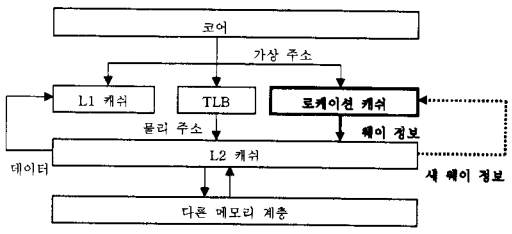

그림 2 로케이션 캐쉬 시스템 [6]

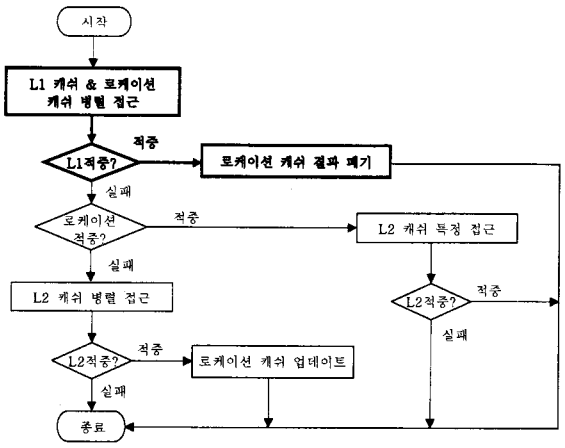

그림 3 로케이션 캐쉬 시스템의 알고리즘

얼마나 많은 웨이 정보가 폐기 되는지 알아보기 위해 표 1 에서 L1 캐쉬의 적중률을 나타내었다. 이 실험은 Rui Min 등의 로케이션 캐쉬 시스템 [6] 에서의 실험 환경과 유사하게 구성하였다. 비록 논문 [6] 에서 Spec2000을 사용한 것과 달리 SpecInt95를 사용하였지만, 캐쉬 구성 환경은 동일하다. 명령 캐쉬 및 데이터 캐쉬는 각 4-웨이 집합-연관으로, 그리고 라인 크기는 64-바이트로 구성하여 전체 크기는 16-킬로바이트가 된다. 실험 결과, L1 명령 캐쉬와 데이터 캐쉬에서 98% 이상의 적중률을 보였다. 다시 말해, 로케이션 캐쉬 시스템에서는 98% 이상의 로케이션 캐쉬의 결과가 그 정확성을 따져보기도 전에 그대로 폐기 된다는 의미가 되겠다.

4. 스피시픽-웨이 캐쉬 시스템

4.1. 스피시픽-웨이 캐쉬의 구조

스피시픽-웨이 캐쉬는 데이터 영역으로 단 1-바이트 만이 필요하다. 본 논문에서 제안한 스피시픽-웨이 캐쉬의 구조는 그림 4 와 같다.

표 1 SpecInt95 에서의 L1 캐쉬의 적중률

SpecInt95	L1 명령 캐쉬	L1 데이터 캐쉬
compress	99.80%	94.77%
gcc	96.00%	98.37%
go	96.00%	98.10%
ijpeg	99.97%	99.63%
li	99.97%	98.50%
m88ksim	100.00%	98.86%
perl	96.89%	99.24%
vortex	96.10%	98.30%
Total average	98.09%	98.22%

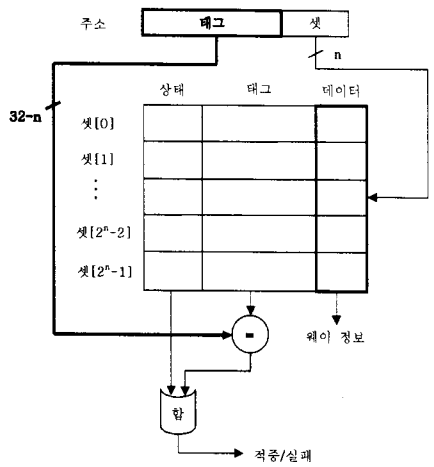

그림 4 스피시픽-웨이 캐쉬

4.2. L2 캐쉬 태그 중복 회피

스피시픽-웨이 캐쉬 시스템은 로케이션 캐쉬 시스템이 안고 있는 L2 캐쉬의 태그 중복 문제를 회피하였다(그림 5). 웨이 정보 추출 방법으로 로케이션 캐쉬 시스템에서는 L2 캐쉬의 중복 태그를 이용해야 했지만, 스피시픽-웨이 캐쉬 시스템은 L2 캐쉬의 태그로부터 웨이 정보가 바로 추출되는 구조를 갖고 있기 때문이다. 따라서 L2 캐쉬의 태그 중복을 회피하여 웨이 정보를 출력함으로써, 추가의 비교기 역시 필요하지 않게 된다.

4.3. 개선된 스피시픽-웨이 캐쉬 접근

또한, 스피시픽-웨이 캐쉬 시스템은, 로케이션 캐쉬 시스템에서의 사용되지 않은 채 폐기되는 로케이션 캐쉬의 접근 방식을 개선하였다. 로케이션 캐쉬가 L1 캐쉬와 병렬적으로 접근되었던 반면, 스피시픽-웨이 캐쉬는 그렇지 않다는 것을 그림 6 에서 볼 수 있다.

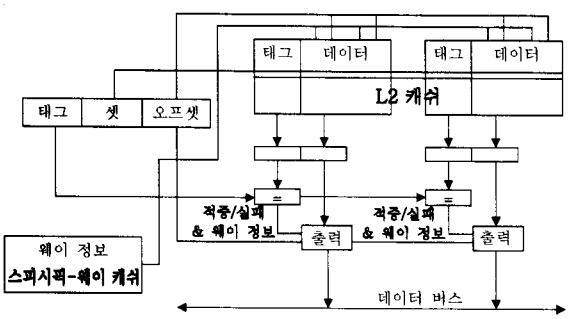

그림 5 스피시픽-웨이 캐쉬 시스템의 블록다이어그램

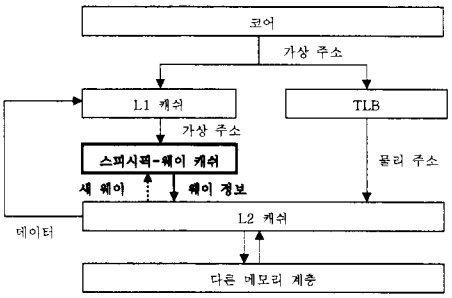

그림 6 스피시픽-웨이 캐쉬 시스템

그림 7 에서는 스피시픽-웨이 캐쉬 시스템의 알고리즘을 도식화 하였다. L1 캐쉬가 접근할 때마다 로케이션 캐쉬도 접근되어야 했던 로케이션 캐쉬 시스템과는 달리, 스피시픽-웨이 캐쉬 시스템에서는 L1 캐쉬가 실패될 때에만 스피시픽-웨이 캐쉬를 접근하여 실제 사용되지 않는 캐쉬 접근을 최소화 하였다. 그렇지만, 스피시픽-웨이 캐쉬는 L1 캐쉬와 병렬적으로 접근하지 않음으로 인해, 시간 지연 문제가 발생된다는 단점이 생기게 된다. 이러한 장단점은 차후에 더 생각해 볼 문제로 남는다.

5. 성능 실험

본 논문에서는 스피시픽-웨이 캐쉬의 예상-적중률(prediction-hit rate)을 실험 대상으로 하였다. 예상-적중률이란 스피시픽-웨이 캐쉬에서 출력한 웨이 정보를 바탕으로, L2 캐쉬의 특정 웨이로 접근한 후의 적중률을 의미한다. 예상-적중률은 Rui Min의 로케이션 캐쉬 [6] 에서의 예상률(prediction rate)과 다르다. L1 캐쉬의 접근시마다 매번 함께 접근되었던 로케이션 캐쉬와는 달리, 스피시픽-웨이 캐쉬는 L1 캐쉬가 실패할 때에만 접근되기 때문이다. 따라서 스피시픽-웨이 캐쉬에서의 예상-적중률이 로케이션 캐쉬의 예상률에 비교하여 볼 때, 보다 실제적인 수치가 되겠다.

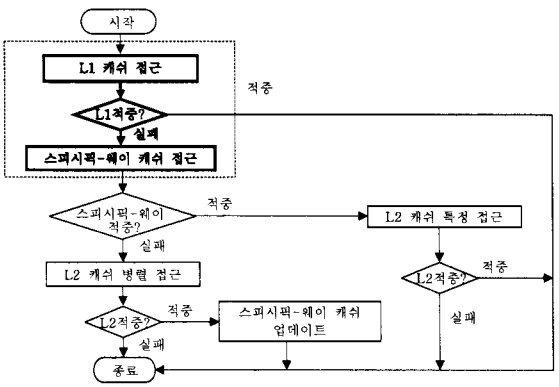

그림 7 스피시픽-웨이 캐쉬 시스템의 알고리즘

표 2 벤치마크 구성

캐쉬	라인 크기 (바이트)	연관 정도	전체크기 (바이트)
L1 명령 캐쉬	16	1	256
L1 데이터 캐쉬	16	1	256
L2 명령 캐쉬	16	4	2048
L2 데이터 캐쉬	16	4	2048
스피시픽-웨이 캐쉬	1	1	32~512

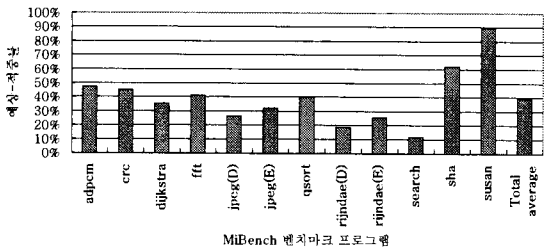

그림 8 스피시픽-웨이 캐쉬의 예상-적중률

본 논문에서 제안한 스피시픽-웨이 캐쉬 시스템을 시뮬레이션 하기 위해 Simplescalar 3.0 [8][9] 을 사용하였고 MiBench [10] 를 이용하여 예상-적중률을 측정하였다. 또한 명령 캐쉬를 제외한 데이터 캐쉬만을 대상으로 하였다. 앞서 언급했듯, 스피시픽-웨이 캐쉬의 접근은 L1 캐쉬의 실패를 전제로 한다. 따라서 상대적으로 작은 크기의 L1 캐쉬로 구성하여 시뮬레이션 하였다. L1 명령 및 데이터 캐쉬는 각 256-바이트의 직접-사상 캐쉬로 구성하였고, L2 명령 및 데이터 캐쉬는 FIFO(First In First Out) 를 대체 방식으로 하는 2-킬로바이트의 4-웨이 집합-연관 캐쉬로 구성하였다. 스피시픽-웨이 캐쉬는 크기에 따라 32-바이트에서 512-바이트까지 다양하게 구성하였다(표 2).

MiBench를 이용한 벤치마크의 결과 평균 39.6% 의 예상-적중률을 나타내었고, 예상의 영향을 가장 많이 받는 벤치마크는 susan인 것으로 나타났다.

6. 결론

집합-연관 캐쉬는 직접-사상 캐쉬보다 적중률이 높다는 장점이 있는 반면, 전력 소모가 많다는 단점이 있다. 그러한 단점을 줄이기 위해 웨이-프리딕팅 셋-어소시에이티브 캐쉬, 로케이션 캐쉬 시스템 등의 연구들이 계속 되어왔다. 본 논문에서는 로케이션 캐쉬 시스템에서 발생하는 L2 캐쉬의 태그 및 비교기 중복, 그리고 로케이션 캐쉬의 사용되지 않은 채 폐기되는 접근 방식을 개선하여, 보다 효율적인 스피시픽-웨이 캐쉬 시스템을 제안하였다. 또한 Simplescalar 3.0 을 이용하여 평균 39.6%의 예상-적중률을 측정하였다.

참고문헌

[1] J. Hennessy and D. Patterson, *Computer Architecture, a quantitative approach*, 3rd Edition, Morgan Kaufmann, San Francisco, CA, 2003.

[2] C. L. Su and A. M. Despain, "Cache Design Trade-offs for Power and Performance Optimization: A Case Study," *Proc. Int'l Symp. on Low Power Electronics and Design*, pp. 69-74, Apr. 1995.

[3] Hasegawa, A., et al., "SH3: High Code Density, Low Power," *IEEE Micro*, pp. 11-19, Dec. 1995.

[4] M. Powell, A. Agrawal, T. Vijaykumar, B. Falsafi, and K. Roy, "Reducing Set-Associative Cache Energy via Way-Prediction and Selective Direct-Mapping," *Proc. Int'l Symp. on Microarchitecture*, pp. 54-65, Dec. 2001.

[5] K. Inoue, T. Ishihara, and K. Murakami, "Way-Predicting Set-Associative Cache for High Performance and Low Energy Consumption," *Proc. Int'l Symp. on Low Power Electronics and Design*, pp. 273-275, Aug. 1999.

[6] Rui Min, Wen-Ben Jone, and Yiming Hu, "Location Cache: A Low-Power L2 Cache System," *Proc. Int'l Symp. on Low Power Electronics and Design*, pp. 120-125, Aug. 2004.

[7] Cheng Shi-You and Huang Juinn-Dar, "Low Power Instruction Cache Architecture Using Pre-Tag Checking," *Proc. Int'l Symp. on VLSI Design, Automation and Test*, pp. 1-4, Apr. 2007.

[8] D. Burger and T. Austin, "The Simplescalar Tool set, Version 2.0," Technical Report #1342, Dept. of Computer Science, Univ. of Wisconsin, Madison, 1997.

[9] T. Austin, E. Larson, and D. Ernst, "Simplescalar: An Infrastructure for Computer System Modeling," *IEEE Computer*, vol.35 no.2, pp. 59-67, Feb. 2002.

[10] M. R. Guthaus et al. "MiBench: A free, commercially representative embedded benchmark suite," *Proc. of the Fourth IEEE Workshop Workload Characterization*, pp. 10-12, Dec. 2001.