데이터 캐시의 활용도를 높이는 동적 선인출 필터링 기법

전영숙 [0], 이병권, 김석일, 전중남
충북대학교 컴퓨터 과학과
yschon@hanmail.net

Dynamic Prefetch Filtering Schemes to Enhance Utilization of Data Cache

Young-Suk Chon[0], Byung-Kwon Lee, Sukil Kim and Joongnam Jeon
Dept. of Computer science, Chungbuk National University

요 약

캐시 선인출 기법은 메모리 참조에 따른 지연시간을 줄이는 효과적인 방법이다. 그러나 너무 적극적인 선인출은 캐시 오염을 유발시켜 선인출에 의한 장점을 상쇄시킨다.

본 연구에서는 캐시의 오염을 줄이기 위해 동적으로 필터 테이블을 참조하여 선인출 명령을 수행할 지의 여부를 결정하는 4가지 필터링 방법들을 비교 평가한다. 비교 연구를 위한 이상적인 필터링 구조를 제안하였으며, 기존 연구에서의 잠김 현상을 개선하기 위한 이진 상태 구조를 제안하였다. 또한, 정교한 필터링을 위한 블록주소 참조 방식을 제안하였다.

일반적으로 많이 사용되는 일반 벤치마크 프로그램과 멀티미디어 벤치마크 프로그램들에 대하여 실험한 결과, 캐시 미스율이 이진 상태 구조는 평균 5.6%, 블록주소 참조 구조는 7.9% 각각 감소하였다.

1. 서론

캐시 구조의 선인출 기법은 프로세서가 사용할 것으로 예측되는 데이터를 실제로 필요하기 전에 주기억장치에서 캐시로 인출하여 캐시 미스율을 감소시키는 방법이다. 캐시 선인출 방식에는 OBL 방식과 RPT 방식 그리고 correlation 방식 등이 있다. 이러한 하드웨어 캐시 선인출 방식들은 메모리 참조 지연시간을 줄이는 효과적인 방법이지만, 너무 과도한 선인출의 사용은 비 효율적인 선인출에 의해 캐시 안의 유용한 데이터를 교체하게 되며(캐시 오염), 버스 트래픽을 증가시켜 전반적으로 성능을 감소시킨다.

비 효율적인 선인출을 줄이기 위해 Zhuang등은 동적 필터링 방법을 제안하였으며 이를 이용하여 불필요한 선인출의 수를 매우 감소시켰음을 보였다[1]. 그러나 이 방법은 선인출 수에만 중점을 두었고, 전체 성능 면에서는 고려하지 않았다. 즉, 성능을 저하시키지 않고 선인출 수를 감소시키기 위해서는 유용한 선인출은 보존되면서 불필요한 선인출만 감소시켜야 되는데, Zhuang등에 의한 방법은 유용한 선인출과 불필요한 선인출 모두를 감소시켰다.

본 논문에서는 이러한 문제점을 해결하기 위해, 유용한 선인출은 최대한 보존하고 불필요한 선인출만을 선택적으로 감소시켜 전체 성능을 향상시킬 수 있는 필터링 방법을 제안하고자 한다. 이를 위해서 세가지 새로운 필터링 구조를 제안하여 기존 방법과 비교하였다.

본 논문의 구성은 다음과 같다. 2 장에서는 본 논문의 대상인 동적 선인출 필터에 대하여 설명한다. 3 장에서는 여러 가지 필터 구조에 대해 자세히 설명한다. 4 장에서는 실험을 통한 성능 평가 결과를 제시한다. 마지막으로 5 장에서는 본 논문에서 얻은 결과를 정리한다.

2. 관련 연구

과도한 선인출을 줄이기 위한 방법으로 선인출 필터를 사용할 수 있다. 그림1과 같이 선인출한 데이터가 실제로 사용되었는지 여부를 해당 데이터가 캐시로부터 빠져나갈 때마다 검사하여 이 정보를 필터로 피이드백하여 동적 필터링을 수행한다. 동적 필터 내부에는 필터링 테이블이 존재하여 선인출시에는 선인출기에 의해 개시된 선인출 주소를 테이블로부터 참조(lookup)하여 필터링 조건에 만족하지 않는 경우에 선인출 큐에 저장한다. 또한 동적 최적화를 위하여 캐시에서 데이터가 빠져나갈 때마다 필터 테이블을 적절하게 갱신한다.

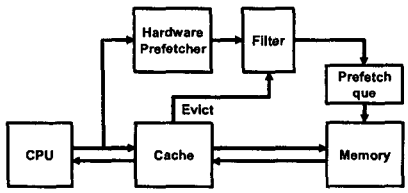


그림 1. 필터링을 적용하는 하드웨어 선인출 구조

3. 선인출 필터링 방식

3.1 이상적인 구조 (Ideal)

가장 이상적인 필터링 방식은 그림2와 같이 선인출하였던 모든 주소에 대하여 선인출 된 이후부터 캐시에서 빠져나갈 때까지 해당 데이터가 사용되었는지 여부를 모두 필터링 테이블에 저장하여 가지고 있다가 이후에 또 다시 동일 주소의 데이터가 선인출 되려고 할 때 테이블을 참조하여 선인출하였으나 사용이 안된 적이 있는 경우에는 선인출 명령을 수행하지 않는다.

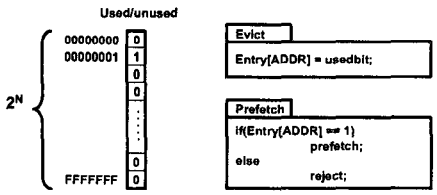


그림 2. 이상적인 필터 구조(Ideal)

3.2 블록주소 참조 구조(Block Address Lookup)

N:1 매핑 구조를 사용하는 기존 동적 방식에서는 정확성이 결여되는 문제가 있다. 그림3과 같이 N:1 매핑 구조의 테이블 크기와 같은 크기를 가지면서 주소를 같이 저장하여 1:1 매핑 구조가 되게끔 하는 방식을 제안한다.

3.3 동적 필터 구조(2bit saturation counter 사용)

Zhuang등에 의해 제안된 필터 구조를 그림 4(a)에 나타내었다. 이 방식에서는 선인출 주소를 일종의 해시 함수를 통과시키는데, 즉 선인출 주소 중에 일부인 하위 비트만을 취하여(캐시라인 주소에 해당하는) 필터링 테이블에 인덱스를 한다. 필터링 테이블의 엔트리는 2bit로 구성이 되어 있으며 캐시에서 빠져나올 때마다 그림 4(b)처럼 상태를 변경시킨다. 선인출기로부터 개시된 선인출 주소의 캐시라인 주소로 테이블을 참조하여 해당 엔트리의 상태값이 2 이상인 경우에 선인출 명령을 수행하고, 그렇지 않은 경우에는 선인출을 하지 않는다.

이러한 기존 동적 필터 구조의 문제점은 두가지로 요약될 수 있다. 첫째로는 캐시라인 주소를 이용한 N:1 매핑에 따른 정확성이 결여된다는 점이다. 둘째로는 2bit 포화 카운터를 사용하기 때문에 결과적으로 불필요한 선인출과 유용한 선인출 모두를 억제시켜 캐시 미스율을 증가시키게 된다.

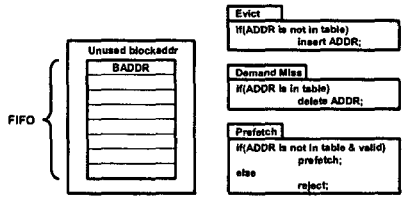
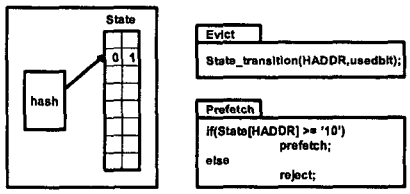
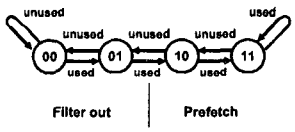


그림 3. unused 블록주소를 갖는 필터 구조(Unused)



(a) 2bit saturation counter를 사용한 필터 구조



(b) 상태도

그림 4. 동적 필터 구조(Prev)

3.4 이진 선택 구조(1bit state 사용)

위와 같은 문제를 해결하기 위해서 그림 5와 같이 카운터를 사용하지 않고 하나의 상태 비트만을 두어서 사용 미사용 여부에 따라서 즉각적으로 선인출 여부가 바뀌도록 하였다.

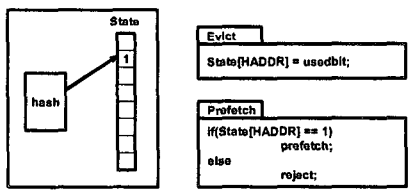


그림 5. 이진 상태를 갖는 필터 구조(Revive)

4. 시뮬레이션 및 성능 분석

4.1 시뮬레이션 환경

필터링을 적용한 선인출 데이터의 효과를 분석하기

위하여 트레이스 구동 시뮬레이션을 수행하였다. 트레이스는 메모리 참조 명령어에 대하여 적재 혹은 저장인지 여부, 그리고 필요한 피연산자의 유효주소로 구성되어 있다. 캐시 시뮬레이터는 Dinero Ⅲ를 기반으로 선인출 알고리즘(OBL, RPT, correlation)과 필터링 방식들을 추가하여 사용하였다. 캐시 시뮬레이터는 캐시크기(4K~1M), 블록크기(32byte), 사상함수(Direct mapped) 그리고 교체 알고리즘(LRU), 필터링 테이블 크기(cache setsize)등의 매개변수를 입력으로 캐시 분석 결과를 생성한다. 대표적인 멀티미디어 벤치마크 프로그램인 MPEG(mpeg2enc, mpeg2dec)과 SpecInt2000의 일반 벤치마크 프로그램인 parser, crafty, vortex, gzip을 대상으로 1천만번의 메모리 참조 명령어에 대하여 실험하였다.

4.2 전체 성능 분석

여러 가지 선인출 알고리즘과 벤치마크를 가지고 캐시 미스율을 비교한 결과를 그림6에 보였다. 결과적으로 하드웨어 선인출기에 의해 선인출이 개시되는 개수가 가장 적은 correlation의 경우 필터링 방식에 따라서 큰 차이가 나지 않는다. 단지, Prev 방식이 다른 방식들보다 미스율이 미약하게 큰 것을 알 수 있다. 다음으로 RPT의 경우에는 Prev 방식은 가장 미스율이 높고, 그 다음으로 Revive방식이 약간 높으며, 나머지 방식들은 비슷한 결과를 보인다. OBL의 경우에는 벤차마크에 따라서 crafty와 mpeg2dec의 경우에는 필터링한 결과가 하지 않은 결과에 비해 모두 더 낮은 미스율을 보인다. 평균적으로 Prev방식을 기준으로 Revive방식은 5.6%, Unused 방식은 7.9% 미스율이 감소하였다.

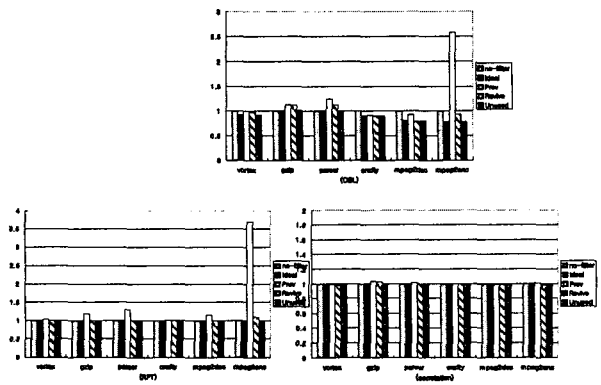


그림 6. 캐시 미스 수 (d128K, b32,direct-mapped)

5. 결론

본 논문에서는 기존의 필터링 구조가 갖는 문제점을 보완하여 여러 가지 필터 구조들을 제안하였다. Prev방식은 캐시라인 주소로 해싱을 하므로 선인출 한 데이터가 사용이 되지 않고 캐시에서 빠져나갈 경우 그 캐시라인에 해당하는 주소 전체의 선인출이 금지되며, 한번 금지가 된 캐시라인은 빠져 나오기가 어렵게 되는 즉, 잠김 상태가 됨을 시간에 대한 선인출 동작 분석을 통하여 알 수 있었다. 이러한 문제를 해결하기 위해 한번이라도 참조가 될 경우 선인출을 할 수 있도록 한 방법이 Revive 방식이다. 또 다른 방법으로서 해싱을 하지 않는 방식을 제안하였으며, 이 방법은 사용이 되지 않은 블록주소를 필터 테이블에 저장하므로 Prev방식이 N:1 매핑인데 비해 1:1 매핑이므로 더 정확하게 필터링을 하는 장점을 가지게 되며 Ideal 방식과 거의 동일한 성능을 가짐을 보였다. 일반 벤치마크와 멀티미디어 벤치마크 프로그램을 가지고 실험을 한 결과, 캐시 미스율은 모든 선인출 알고리즘과 벤치마크들에 대하여 Prev 방식과 비교하여 평균적으로 Revive 방식이 5.6%, Unused 방식이 7.9% 감소함을 알 수 있었으며, 이러한 결과들을 토대로 good 선인출의 수를 보존하면서 bad 선인출은 감소시키도록 하는 것이 전체 캐시 성능을 향상시킨다는 것을 보였다.

선인출의 정확도를 높이고 캐시의 오염을 방지하기 위해 본 논문에서 이러한 방식들을 응용하여 웹이나 데이터베이스 등 여러 다양한 분야에 확장하여 적용하는 선인출 연구가 수행되어야 할 것으로 생각된다.

참고문헌

[1] T-Fu Chen and J-L Baer, " Effective Hardware-Based data prefetching for High-Performance Processors," *IEEE Trans. Computers*, Vol. 44, No. 5, pp. 609-623, May 1995.

[2] X. Zhuang and H-H S. Lee, " Hardware-based Cache Pollution Filtering Mechanism for Xggressive Prefetches," *in Proc. IEEE Int. conf. on Parallel Processing*, pp.286-293, Oct. 2003.

[3] V. Srinivasan, E. S. Davidson and G. S. Tyson, " A Prefetch Taxonomy" , *IEEE Trans. Computers*, Vol. 53, No. 2, pp. 126-140, Feb. 2004.