Journal of Knowledge Information Technology and Systems

ISSN 1975-7700

http://www.kkits.or.kr

Bypassing Scheme for Inclusive Last Level Caches

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

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

ABSTRACT

The design of an effective last level cache(LLC) continues to be an important issue in processor's performance. Recent works on high performance caches have shown that cache bypassing is an effective technique to enhance the performance of last level caches. However, commonly used inclusive cache hierarchy cannot benefit from this technique because bypassing inherently breaks the inclusion property. This paper presents a solution to enabling cache bypassing for inclusive caches. It introduces a bypass buffer to an LLC. Bypassed cache lines skip the LLC while their entries are allocated into the bypass buffer. And it is a simple, low-hardware overhead, yet effective, cache bypassing scheme that dynamically chooses which blocks to insert into the LLC and which blocks to bypass it based on past access/bypass patterns on a miss. Our proposed scheme is evaluated using a detailed simulation environment where its effectiveness, performance-improvement capabilities, and robustness are demonstrated. We present experimental results showing IPC(Instruction Per Cycle) comparison of the proposed scheme and OBM(Optimal Bypass Monitor) against LRU for SPEC CPU2006 benchmarks. The result show that the proposed scheme and OBM can improve IPC by an average of 18% and 14%, respectively. And the proposed scheme reduces the miss rate by 16% compared to LRU.

KEYWORDS: LLC(last-level cache), Cache bypassing, Inclusive caches, Exclusive caches, Bypass buffers

ARTICLE INFO: Received 23 March 2016, Revised 11 April 2016, Accepted 11 April 2016.

*Corresponding author is with the Department of Computer Science, University of Suwon, 17 Wauangil Bongdam-Eup Hwaseong-Si, Gyeonggi-Do, 18323, KOREA. *E-mail address*: yicho@suwon.ac.kr

1. 서 론

효율적인 LLC(Last Level Cache) 설계는 프로세서 성능에 중요하기 때문에 지속적인 연구의 중심에 있다. 그러나 고성능 프로세서에서 LLC는 효율적으로 사용되지 못하고 있으며 여러 연구에서 기존 캐시 설계의 효율성을 지적하였다[1-4]. SPEC CPU2006 벤치마크 중에서 약 50%정도가 LLC로 가져온 블록의 90% 이상이 퇴출되기 전에 다시 액세스 되지 않는다는 것을 보여주었다[5-9]. 재 액세스되지 않는 블록들은 LLC에 있는 동안 캐시 적중에 기여하지 못한다. 만약 재 액세스되지 않는 블록들이 캐시로 삽입되어 더 유용한 캐시 블록들을 대신 퇴출시킨다면 해당 블록들이 불필요하게 캐시 공간을 차지하여 캐시 오염과 스래싱(thrashing)의 원인이 된다. LLC에서 재 액세스되지 않는 주된 이유는 상위 레벨 캐시(L1/L2)에 의해 지역성이 여과되기 때문이다. 즉, 상위 레벨에서 적중되기 때문에 LLC에서는 액세스가 발생하지 않는다.

재 액세스되지 않는 블록 문제에 대한 효과적인 해법이 캐시 바이패싱(cache bypassing)이다. 캐시 바이패싱은 LLC에 있는 동안 재 액세스되지 않는 블록들을 식별하여 미스 때 LLC를 바이패스하고 위 레벨 캐시에만 삽입시킨다. 캐시 바이패싱 방법은 블록의 주소나 미스의 원인이 된 명령어의 주소로 캐시 블록들을 식별하는 부류[7-10]와 블록들의 동작을 실행시간에 학습하여 바이패싱을 결정하는 부류[11-13]가 있다.

캐시 바이패싱은 재 액세스가 기대되는 블록들만 LLC에 저장함에 의해 캐시 용량을 좀 더 효율적으로 이용할 수 있게 하고, LLC에 대한 액세스 대역폭 감소로 인해 성능 이득을 기대할 수 있다. 그러나 기존의 캐시 바이패싱 방법은 여러 단점을 갖는다. 어떤 바이패싱 방법은 테이블을 빈번히 액세스한다는 점과 여러 구조와 필드들의 처리를 요구하는 등 매우 복잡하다[7-10]. 다른 바이패싱 방법은 하드웨어 단가가 비싸다[11-13]. 즉 큰 테이블과 캐시라인 당 태그배열에 많은 추가 필드를 요구한다.

이들 바이패싱 방법들은 배타적 캐시(exclusive cache)에만 적용할 수 있고 포괄적 캐시(inclusive cache)에는 적용 못한다. 배타적 캐시는 여러 캐시 레벨 간에 블록을 이동하는 동안 부과되는 큰 대역폭 압박 때문에 선호되지 않는다. 이전의 캐시 바이패싱 방법은 어떤 특성들을 기초로 하여 캐시 블록들을 여러 그룹으로 나누고 동일 그룹에 있는 블록들은 비슷하게 취급한다. 이것은 동일 그룹 내에 있는 여러 블록들이 연관이 없다면 잘못된 예측을 유발할 수도 있다. 이와 같은 단점들은 개선된 캐시 바이패싱 방법의 개발을 필요로 하고 그것이 본 논문의 주목표이다.

본 논문은 포괄적 캐시를 위한 간단하지만 효율적인 새로운 캐시 바이패싱 방법을 제안한다. 이를 위해 바이패스 버퍼(Bypass Buffer)를 LLC에 추가한다. 바이패스될 캐시 라인들은 LLC를 스킵하고 바이패스 버퍼에 엔트리를 할당한다. 개개 캐시 블록의 과거 LLC 액세스/바이패스 패턴에 기초하여 미스 후에 어떤 블록을 바이패스하고 어떤 블록을 LLC로 삽입할 지를 동적으로 선택한다. 제안한 방법은 약간의 스토리지 오버헤드와 간단한 알고리즘을 사용하여 구현된다.

LLC를 갖는 프로세서에서 평가할 때 제안 방법은 베이스 LRU에 비해 LLC 성능에 종속적인 10개 SPEC CPU2006 벤치마크를 최대 75%, 평균 18% 속도 향상 시킨다.

2. 연구 동기

최근 발표된 캐시 바이패싱 방법[9]은 바이패스될 블록을 LRU(Least Recently Used) 위치에 삽입

함에 의해 포괄적 캐시에서도 동작하도록 하였다. 이 방법에서 선택된 바이패싱 대상자들은 해당 캐시 셋에서 다음 미스 때 희생자가 된다. 그러나 이 방법은 몇 가지 문제점을 갖는다. 첫째, 블록들을 캐시 셋에 저장해야하므로 더 유용한 블록이 교체될 수 있다. 둘째, 연속적인 액세스들이 한 캐시 셋으로 사상되는 경우에 블록들이 장래에 재사용되지 않을 것으로 예측되면 그들은 바이패싱 대상자이므로 LRU 위치를 위해 경쟁할 것이다. 결과적으로 이들 블록의 생존시간은 짧아지고 이는 상위 레벨로부터 동일 블록들이 퇴출되는 원인이 되므로 포괄적 LLC의 성능을 감소시킨다.

캐시 바이패스의 목표는 현재 캐시에 있는 블록들보다 적은 재사용을 갖는 블록들을 바이패스 시키는 것이다. 캐시 바이패싱으로 부터 얻을 수 있는 성능 이득 가능성을 알아보기 위해 바이패싱의 잠재적 성능 개선의 상한을 알아본다. <그림 1>은 LRU 교체 알고리즘에 최적 바이패싱을 적용 시 LLC 미스 감소율을 보여준다.

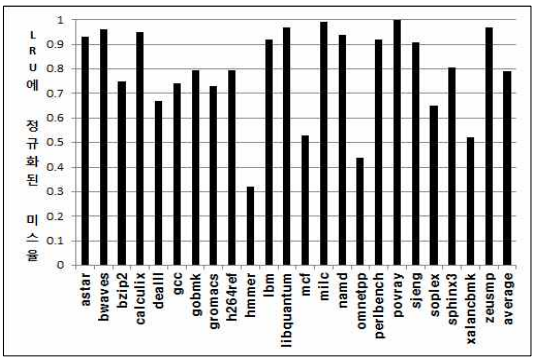

그림1. 최적 바이패싱을 사용시 LRU 교체 정책에 표준화된 LLC 미스율
Figure 1. LLC miss rate of optimal bypassing normalized to LRU replacement policy.

최적 바이패싱은 들어오는 블록의 미래 재사용 거리가 퇴출자의 미래 재사용 거리보다 크면 들어오는 블록을 바이패스 시킨다. 그림은 LRU에 최적 바이패싱을 적용했을 때 캐시미스를 평균 21% 제거시키므로 바이패싱이 미스의 상당 부분을 보상해 줌을 보여준다. 또한 최적 바이패싱은 벤치마크들에서 들어오는 블록의 평균 77%를 바이패스시킨다. 이런 높은 바이패스 율은 최적 바이패싱이 블록들을 적극적으로 바이패스시킴을 알 수 있다.

캐시미스 시 어떤 블록을 바이패스할지를 결정하는 것이 중요하다. 이를 위해 벤치마크들에서 바이패스된 블록들의 생존시간 정보를 분석하였다. 블록의 생존시간은 어떤 캐시 블록이 L1 캐시에서 있는 동안 LLC 미스 수로 측정한다. 즉, 바이패스된 블록이 L1캐시에 할당된 시간과 퇴출되기 직전 사이에 LLC 미스 수로 나타낸다.

<그림2>는 바이패스된 블록들이 L1 캐시에서 빠르게 dead 됨을 보여준다. 예를 들어, ammp는 바이패스된 모든 블록이 0의 생존기간을 가지며, 벤치마크 art는 바이패스된 블록의 78%가 3~4 LLC 미스의 짧은 생존시간을 갖고, 97%가 8 LLC 미스 후에 dead한다. 모든 벤치마크의 평균에서 바이패스된 블록의 95.5%가 8 LLC 미스 후에 dead된다.

L2 캐시에서도 바이패스된 블록들의 생존시간 정보를 분석하였는데 매우 비슷한 경향을 나타낸다.

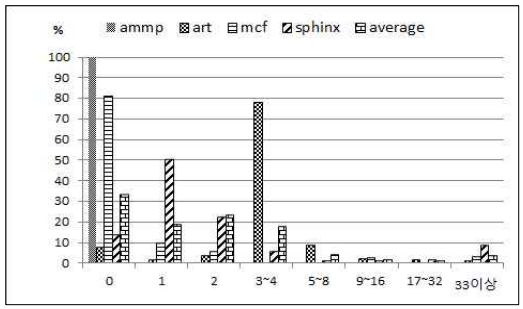

그림2. LLC에서 바이패스된 블록의 L1 데이터 캐시에서의 생존시간
Figure 2. Lifetime of blocks to be bypassed from LLC in L1 data cache.

블록들을 재 액세스 기간에 기초하여 분류할 수 있다. 재 액세스 기간이 짧은 블록은 L1/L2 캐시에서 모두 적중하므로 L1/L2 캐시에 의해 제공될 수 있는 블록이다. 재 액세스 기간이 긴 블록은 모든 캐시 레벨에서 캐시 미스를 유발하는 블록이다. 재 액세스 기간이 중간인 블록은 작은 용량을 갖는 L1/L2 캐시에서는 미스이지만 큰 LLC에서는 적중하는 블록이다. 따라서 재 액세스 기간이 중간인 블록들만 LLC에 있어야하고 재 액세스 기간이 짧거나 긴 블록은 LLC로 삽입하지 말고 바이패스 시켜야한다.

따라서 효과적인 캐시 바이패싱 방법을 설계하기 위해서는 캐시 블록들의 재 액세스 간격을 식별하여 삽입/바이패스 하도록 설계해야한다.

3. 포괄적 캐시를 위한 바이패싱 방법

포괄적 LLCs에서 바이패싱을 가능하도록 하기 위해 바이패스 버퍼(Bypass Buffer, BB)를 추가한다. 바이패스된 블록들은 LLC에 있는 희생자들과 교체하는 대신에 BB에 할당된다. 만약 어떤 블록이 BB로부터 퇴출된다면 포괄적 성질을 만족하기 위해 상위 레벨에 있는 동일 데이터들도 무효화시킨다.

<그림 3>은 제안한 바이패싱 방법의 동작을 기술한다. 각 캐시 라인은 단일 비트(accessed)가 확장되었는데 이 비트는 미스 뒤에 블록이 캐시로 삽입될 때 'false'로 설정되고, 캐시에 있는 동안 그 블록이 재 액세스되면 'true'로 설정된다. 임의 블록이 교체를 위해 선택될 때, accessed 비트는 그 블록이 캐시에 있는 동안 재 액세스됐는지 혹은 아닌지를 지시한다. 또한 제안 방법은 개개 블록들의 액세스/바이패스 히스토리를 저장하기 위해 바이패스 버퍼의 각 엔트리에 2비트 포화 카운터(cnt)가 할당된다. 카운터는 벤치마크들에서 들어오는 블록의 평균 77%가 바이패스 된다는 분석결과를 적용한다. 블록 B가 LLC 미스일 때 BB에서 블록 B의 카운터를 조사한다.

```
Algorithm Cache_Bypass
begin
   access a block B
   if block B is hit on LLC then B.accessed = true;
   else { // LLC miss
      if block B is in BB then {
         if BB(B).cnt < 3 { // bypass
              BB(B).cnt++;
              bypass B;
         else { // BB(B).cnt == 3, replacement
            find a victim block V using LRU;
            insert B in LLC;
            B.accessed = false;
            allocate V into BB;
            if V.accessed == true then BB(V).cnt = 3;
                                  else BB(V).cnt = 0;
         }
      else   // B is not in BB and LLC
         allocate B into BB; BB(B).cnt = 0;
   }
end.
```

그림3. 제안한 바이패싱 방법
Figure 3. Proposed bypassing scheme

카운터 값이 3보다 작으면 블록 B는 LLC에 있는 동안 재 액세스되지 않을 것으로 예측하여 바이패스 시킨다. 그리고 블록 B의 카운터를 증가한다. 만약 블록 B의 카운터 값이 3이면 해당 블록은 액세스 빈도가 높으므로 LLC로 삽입한다. 이때 교체될 희생자 블록 V를 BB에 추가한다. 희생자 블록 V의 카운터는 블록 V가 캐시에 있는 동안 액세스됐는지(카운터를 3으로 셋) 아닌지(카운터를 0으로 리셋)에 따라 설정한다.

BB는 4K 2비트-엔트리를 갖는 직접 사상(direct mapping)이고 태그와 데이터를 포함하지 않는다. 따라서 총 사이즈는 약 1KB이므로 1MB LLC의 경우 추가 오버헤드는 0.1%이다. 또한 BB는 LLC 미스 때만 액세스되고, 액세스 시간은 LLC 미스 지

연(latency)과 완전히 중첩된다. BB 인덱스를 위해 블록 주소가 12 비트 부분들로 나뉘어 XOR된다.

각 LLC 캐시 라인은 1비트 accessed 필드가 확장된다. 64B 라인을 갖는 1MB LLC의 경우 accessed 필드를 위한 총 오버헤드는 2KB이고, 이는 캐시 크기의 0.2%이다. 제안한 바이패싱 방법은 구현이 간단하고, 매우 적은 하드웨어 오버헤드를 갖지만 LLC의 성능과 효율성을 상당히 개선할 수 있다.

4. 실험환경

제안한 바이패싱 방법은 [14]에 기초한 실행-구동 시뮬레이터를 사용하여 평가한다. 128-엔트리 ROB와 8-스테이지를 갖는 4-way 비순차적 슈퍼스칼라 프로세서를 모델링하였다.

<표 1>은 베이스라인 캐시 구성을 보여준다. 베이스라인 L1 명령어 캐시는 32KB 4-way 셋 연관(set associative)이고 데이터 캐시는 8-way 셋 연관이다. 베이스라인 L2 캐시는 256KB 8-way 셋 연관이다. L3 캐시는 1MB 16-way 셋 연관이고 베이스라인의 모든 레벨의 캐시는 64B 라인 크기를 사용한다. 표에서 LRU는 기존의 LRU 교체정책을 의미하고, LRU 교체정책을 베이스라인 캐시에서 사용한다.

표 1. 베이스라인 캐시 구성,
Table 1. Configuration of baseline cache

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

제안한 바이패싱 방법을 평가하기 위해 SPEC CPU2006 벤치마크를 사용한다. 본 방법은 효율성과 성능을 개선시키는 구조이기 때문에 LLC의 성능이 실행시간에 크게 영향을 주는 벤치마크에서 영향이 뚜렷하게 나타난다. 23개 SPEC CPU2006 벤치마크 중에서 LLC 크기를 1MB에서 4MB로 증가할 때 최소 10% 이상의 성능이 개선되는, LLC 성능에 종속적인 10개 벤치마크에 대해 평가한다. LLC 크기를 증가할 때 10개 벤치마크의 성능 개선은 <그림 4>에서 보여준다. 이들 벤치마크는 콜드 스타트(cold start) 미스의 영향을 제거하기 위해 준비시간(warm-up)을 갖고 reference 입력을 사용하여 SimPoint[15]로 부터 각 벤치마크에 대해 250M 명령어 트레이스를 얻었다.

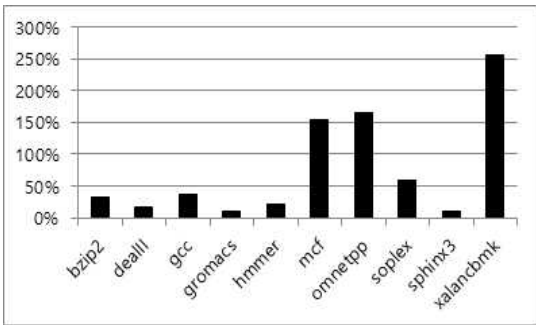

그림 4. LLC를 1MB에서 4MB로 증가시켰을 때 성능 개선
Fig. 4. Performance improvement when the LLC size is increased from 1 MB to 4 MB

5. 실험결과

<그림 5>는 제안한 바이패싱 방법과 [9]에서 제안한 OBM(Optimal Bypass Monitor)에 대해 IPC 성능 개선을 비교하였다. 성능 개선은 캐시 바이패싱이 없는 베이스 캐시 구성에 대한 상대적 값으로 계산되었다.

제안 방법은 LLC에 성능 종속적인 벤치마크에 대해 평균 18%, 최대 75%(mcf) 성능 향상을 얻었

다. 제안 방법은 대부분의 벤치마크에서 5% 이상 성능 향상을 보였다. 평균 성능 향상은 제안 방법이 18%, OBM이 14%으로 제안 방법이 OBM보다 우수한 것으로 나타났다.

제안 방법은 mcf, omnetpp, xalancbmk에서는 OBM보다 상당히 우수하였지만 너무 적극적인 바이패싱으로 인해 sphinx3에서는 OBM보다 약간 성능이 떨어지는 것으로 나타났다.

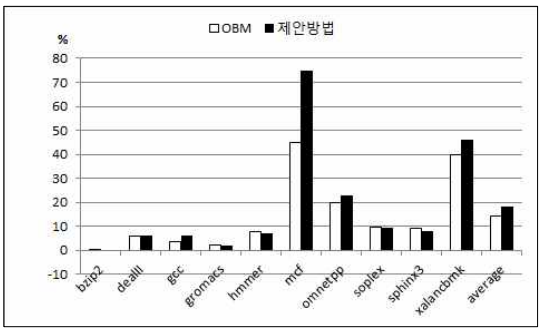

그림 5. 성능 향상
Figure 5. Performance improvement in benchmarks

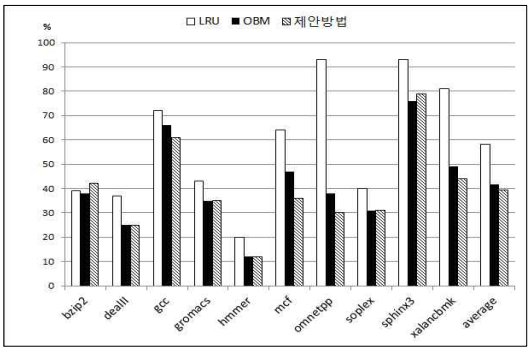

그림 6. LRU, OBM, 제안방법의 미스율
Figure 6. Miss rates of LRU, OBM and proposed scheme

<그림 6>은 OBM과 제안 방법이 캐시에 유용한 블록들을 유지하고 유용하지 않은 캐시 블록들을 바이패스 함에 의해 LRU에 비해 미스 율이 효과적으로 감소하였음을 보여준다. 더 큰 LLCs(2MB, 4MB)에서 제안방법의 효율성과 성능을 평가하고 구현했을 때 각각 15%, 16% 성능 향상을 얻었다.

6. 결 론

고성능 캐시를 위한 최근의 연구들에서 캐시 바이패싱이 LLC 성능을 개선시키는 효과적인 기술이라는 것을 보여주었다. 그러나 바이패싱은 포괄적 성질을 깨트리기 때문에 포괄적 캐시에서는 바이패싱 기술로부터 이득을 얻을 수 없다.

본 논문에서는 포괄적 캐시를 위한 간단하면서 효율적인 캐시 바이패싱 방법을 제안하였다. 제안한 바이패싱 방법은 블록단위로 실행시간에 캐시 블록의 동작을 동적으로 학습하고 BB에 있는 2-비트 포화 카운터에 블록의 학습된 동작을 저장한다. 이 학습된 정보는 캐시 미스 때 삽입할지 혹은 바이패스할 지를 결정하는데 이용된다.

본 바이패싱 방법은 실행 구동 시뮬레이션 환경을 사용하여 평가하였다. 시뮬레이션 결과 제안한 바이패싱 방법은 캐시 성능과 효율성을 상당히 개선시키는 능력을 확인하였다. 또한 제안 방법은 약간 하드웨어와 간단한 알고리즘을 사용하면서도 최신 캐시 바이패싱 방법인 OBM보다 좋은 성능을 보였다.

References

[1] X. Chen, S. Wu, L. Chang, W. Huang, and W. Hwo, *Adaptive cache bypass and insertion for many-core accelerators*, In MICRO-2014.

[2] N. Duong, D. Zhao, T. Kim, R. Cammarota , M. Valelo, and A. Veidenbaum, *Improving cache management policies using dynamic reuse distances*, In MICRO-2012, pp.389-

400, 2012

[3] M. Gebhart, D. Johnson, D. Tarjan, S. Keckler, W. Dally, E. Lindholm, and K. Skadron, *Energy-efficient mechanisms for managing thread context in throughput processors*, In ISCA-2011, pp. 235–246, 2011.

[4] S. Lee, and Y. Cho, *A cache replacement policy for improving the performance of last level cache in processors*, JKITS, Vol. 10, No. 2, pp. 145-152, Apr. 2015.

[5] A. Jaleel, E. Borch, M. Bhandaru, S. Steely, and J. Emer, *Achieving non-inclusive cache performance with inclusive caches - temporal locality aware (TLA) cache management policies*, In MICRO-2010, pp. 151-162, 2010.

[6] A. Jaleel, K. Theobald, S. Steely, and J. Emer, *High performance cache replacement using re-reference interval prediction (RRIP)*, In ISCA-2010, pp. 60-71, 2010.

[7] J. Gaur, M. Chaudhuri, and S. Subramoney, *Bypass and insertion algorithms for exclusive last-level caches*, In ISCA-2011, pp. 81-92, 2011.

[8] S. Gupta, H. Gao, and H. Zhou, *Adaptive cache bypassing for inclusive last level caches*, In IPDPS-2013, pp. 1243-1253, 2013.

[9] L. Li, D. Tong, and X. Cheng, *Optimal bypass monitor for high performance last-level caches*, In PACT-2013, pp. 315-324, 2013.

[10] M. Chaudhuri, J. Nuzman, I*ntroducing hierarchy-awareness in replacement and bypass algorithms for last-level caches*, In PACT-2013, pp. 293-304, 2013.

[11] M. Kharbutli, and Y. Solihin, *Counter-based cache replacement and bypassing algorithms*, IEEE Trans. on Computers, 57(4), pp. 433-447, 2008.

[12] H. Dybdahl, and P. Stenstrom, *Enhancing last-level cache performance by block bypassing and early miss determination*, In ACSAC-2006, pp. 52-66, 2006

[13] C. Li, S. Song, H. Dai, A. Sidelnik, S. Hari, and H. Zhou, *Locality-driven dynamic GPU cache bypassing*, In ICS-2015, pp. 67-77, 2015.

[14] V. Krishnan, and J. Torrellas, *A direct-execution framework for fast and accurate simulation of superscalar processors*, In PACT 1998, pp. 286-293, 1998.

[15] SimPoint home page : http://cseweb.ucsd.edu/~calder/simpoint/, Oct., 2015

포괄적 마지막 수준 캐시를 위한 바이패싱 기법

이상정[1], 조영일[2]

[1]*순천향대학교 컴퓨터공학과*

[2]*수원대학교 컴퓨터학과*

요 약

효율적인 마지막 수준 캐시(last level cache, LLC)의 설계는 프로세서 성능에 중요한 문제가 되고 있다. 고성능 캐시에 대한 최근 연구들은 캐시 바이패싱이 LLC 성능을 향상시키는 효과적인 방법이라는 것을 보여주었다. 그러나 바이패싱은 고유의 포괄(inclusion) 성질을 깨뜨리기 때문에 일반적으로 사용하는 포괄적 캐시는 바이패싱 기술로부터 이득을 얻을 수 없다. 본 논문에서는 포괄적 캐시에서도 캐시 바이패스가 가능한 해법을 제안한다. 제안 방법은 LLC에 바이패스 버퍼를 도입한다. 바이패스된 캐시 라인은 바이패스 버퍼에 엔트리를 할당하고 LLC를 스킵한다. 제안 방법은 과거 액세스/바이패스 패턴에 기초하여, 캐시 미스 때 어떤 블록을 LLC에 삽입하고 어떤 블록을 바이패스할 지를 동적으로 선택하는 간단하고 적은 하드웨

어 오버헤드를 갖지만 효율적인 바이패스 방법이다. 제안 방법은 알고리즘의 효율성, 성능 개선 능력 및 강력함을 증명하기 위해 정밀한 시뮬레이션 환경을 사용하여 평가하였다. 논문에서 SPEC CPU2006 벤치마크에 대해서 LRU에 비해 제안한 방법과 OBM의 성능개선(IPC)을 보여주는 실험 결과를 제시한다. 제안 방법은 평균 18%, OBM은 평균 14% IPC를 개선시킨다. 또한 제안 방법은 LRU와 비해 16% 미스 율을 감소시킨다.

Sang-Jeong Lee received the B.S., M.S., Ph.D. 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 address : sjlee@sch.ac.kr

Young-Il Cho received the B.S., M.S., Ph.D. 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 global sensor network.

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