ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 42, No. 8, pp. 947-953, 2015. 8
http://dx.doi.org/10.5626/JOK.2015.42.8.947

하이브리드 메인 메모리와 스토리지의 특성을 고려한 버퍼 캐시 교체 정책

(A Buffer Cache Replacement Algorithm for Considering both Hybrid Main Memory and Storage)

강 동 현 † 엄 영 익 ††
(Dong Hyun Kang) (Young Ik Eom)

요 약 PRAM은 바이트 단위의 쓰기와 비휘발성의 특징을 모두 가지고 있으며, DRAM보다 높은 밀집도가 기대되기 때문에 DRAM을 대체할 수 있을 것으로 예상된다. 이에, PRAM 기반의 버퍼 캐시 교체 정책에 대한 연구가 활발하게 진행되고 있다. 그러나 대부분의 기존 연구는 PRAM의 수명 및 느린 쓰기 성능에만 집중함으로써 PRAM의 바이트 단위의 쓰기 성능을 제한적으로 이용한다. 이에, 본 논문에서는 PRAM의 바이트 단위의 쓰기 성능과 스토리지의 성능을 모두 고려한 새로운 버퍼 캐시 교체 정책을 제안한다. 제안 기법은 바이트 단위의 쓰기 성능을 이용하기 위해 작은 크기의 쓰기 요청이 빈번한 페이지를 PRAM에 유지시키며 DRAM과 PRAM사이의 선택적 페이지 이동을 통해 PRAM의 쓰기 횟수를 감소시킨다. 실험 결과, 제안 기법은 CLOCK 알고리즘에 비해 최고 92%까지 PRAM의 쓰기 횟수를 감소시키고 PRAM 테스트 보드에서 최대 62%까지 수행시간을 향상시키는 것을 확인하였다.

키워드: 버퍼 캐시 교체 정책, PRAM, 하이브리드 메인 메모리, 스토리지

Abstract PRAM is being considered as a potential successor to DRAM because of its characteristics such as byte-addressability, non-volatility, and high density. To gain its benefits, buffer cache replacement algorithm based on PRAM has been actively studied. However, most of the previous studies on buffer cache replacement algorithm limitedly exploit the byte-level performance of PRAM by focusing its limited lifetime and slower access latency compared to DRAM. In this paper, we propose a novel buffer cache replacement algorithm that fully considers the byte-level performance of PRAM and the performance of secondary storage. To take advantage of small size write on PRAM, proposed scheme keeps pages, which are frequently accessed with a small size write, on PRAM and allows the selective page migration from DRAM to PRAM. As a result, our scheme significantly reduces the number of PRAM writes. Our experimental results indicate for real workloads that our scheme reduces the number of PRAM writes by up to 92% and improves its performance by up to 62% compared to CLOCK.

Keywords: buffer cache replacement algorithm, PRAM, hybrid main memory, storage

· This investigation was financially supported by Semiconductor Industry Collaborative Project between Sungkyunkwan University and Samsung Electronics Co. Ltd
· 이 논문은 제41회 동계학술발표회에서 '페이지 캐시를 위한 바이트 단위의 PCM 성능 분석'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 성균관대학교 정보통신대학
kkangsu@skku.edu
†† 종신회원 : 성균관대학교 정보통신대학 교수(Sungkyunkwan Univ.)
yieom@skku.edu
(Corresponding author임)

논문접수 : 2015년 2월 2일
(Received 2 February 2015)
논문수정 : 2015년 4월 27일
(Revised 27 April 2015)
심사완료 : 2015년 5월 12일
(Accepted 12 May 2015)

정보과학회논문지 제42권 제8호(2015. 8)

1. 서 론

최근 시스템 분야의 학계와 산업계는 높은 집적도, 비휘발성, 그리고 바이트 단위 접근 등의 특징을 가진 PRAM을 주목하고 있다[1-6]. 특히, PRAM은 DRAM처럼 메모리 버스를 통해 바이트 단위의 접근이 가능함에도 불구하고 저장된 데이터를 유지하기 위해 주기적인 전원 공급 (memory refresh)을 필요로 하지 않기 때문에 향후 DRAM을 대체할 것으로 예상되고 있다[2]. 그러나 PRAM은 DRAM에 비해 제한된 수명과 느린 읽기 및 쓰기 성능 등의 단점도 가지고 있어, 기존 연구들은 DRAM과 PRAM을 함께 사용하는 하이브리드 메인 메모리를 소개하고 DRAM과 PRAM의 특성을 고려한 버퍼 캐시 교체 정책을 제안하였다[3-6].

대부분의 하이브리드 메인 메모리 기반의 버퍼 캐시 교체 정책은 DRAM에 비해 제한된 PRAM의 수명과 느린 쓰기 성능을 고려하기 위해, 읽기 요청이 빈번한 페이지(read-intensive page)는 PRAM에 유지하고 쓰기 요청이 빈번한 페이지(write-intensive page)는 DRAM에 유지하는 정책을 이용한다[4-6]. 그 결과, PRAM의 쓰기 횟수를 감소시킴으로써 제한된 PRAM의 수명과 시스템의 성능을 향상시킬 수 있었다. 그러나, 기존의 연구는 빈번하게 접근되는 페이지를 구분하기 위해 DRAM과 PRAM 사이의 페이지 이동을 수행하기 때문에 작은 크기의 읽기 및 쓰기 요청에도 페이지 단위(4 KB)의 쓰기가 DRAM과 PRAM에 각각 요청될 수 있다. 예를 들어, DRAM에 여유 메모리(free memory)가 존재하지 않는 경우, PRAM의 페이지에 요청된 작은 크기의 쓰기를 위해 DRAM의 페이지가 PRAM으로 이동된다. 또한, 버퍼 캐시가 더 이상 여유 메모리을 가지고 있지 않은 경우, 덮어 쓰기(in-place update)을 수행하지 않고 페이지 이동을 수행하는 기존 교체 정책들은 불필요한 페이지 교체를 발생시킨다. 특히, 버퍼 캐시의 교체 페이지가 스토리지에 반영되어야 하는 경우, 스토리지의 쓰기 요청이 발생하는 잠재적인 문제점을 포함하고 있다.

본 논문에서는, 덮어 쓰기를 수행하지 않고 페이지의 이동을 수행함으로써 페이지 크기의 쓰기를 요청하는 기존 하이브리드 기반의 버퍼 교체 정책들이 실제 PRAM의 쓰기 성능과 어떠한 관계가 있는지 확인하기 위해, ARM 기반의 PRAM 테스트 보드에서 읽기 및 쓰기 요청 크기에 따른 실제 PRAM의 성능을 비교해 보았다. 성능 평가는 PRAM의 읽기 및 쓰기의 요청 양을 2 Byte부터 4 KB까지 점차 증가시키는 방식으로 수행되었으며 동일한 크기의 읽기 및 쓰기 요청을 10000번 수행했을 때의 소모 시간을 비교하였다. 즉, 2 바이트의 쓰기 성능 결과는 2 바이트 크기의 쓰기 요청을 임의의 PRAM 주소에 10000번 수행한 후의 전체 소모시간을 의미한다. 그림 1은 실제 PRAM의 읽기 및 쓰기 성능을 비교하여 보여준다. 그림 1에서 보여주듯이, 실제 PRAM의 성능은 PRAM에 요청하는 양이 증가할수록 점차 증가하는 것을 확인할 수 있다. 특히, 쓰기 요청 양에 따른 PRAM의 성능 차이는 4KB 쓰기 요청 결과를 기준으로 2배에서 최고 32배까지 차이를 보였다. 이러한 성능 결과는 하이브리드 메인 메모리로 PRAM을 이용하는 경우, PRAM에 요청되는 양이 시스템의 전체 성능에 중요한 영향을 미친다는 사실을 보여준다.

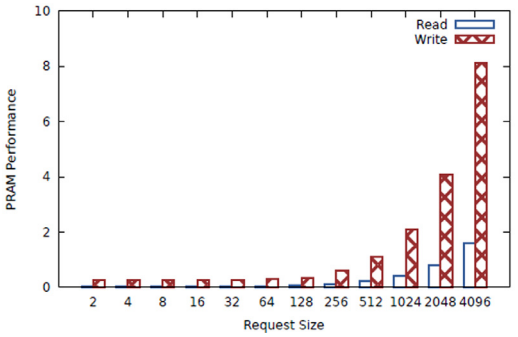

그림 1 PRAM의 읽기 및 쓰기 성능
Fig. 1 Read and Write Performance of PRAM

이에, 본 논문에서는 하이브리드 메인 메모리 환경에서 PRAM의 쓰기 요청 양에 따른 성능 차이를 고려하는 새로운 버퍼 캐시 교체 정책을 제안한다. 제안 기법은 작은 크기의 쓰기 요청이 빈번한 페이지를 PRAM에 유지시키고 큰 크기의 쓰기 요청이 빈번한 페이지를 DRAM에 유지시킴으로써 시스템의 전체 성능을 향상시킨다. 특히, 제안 기법은 PRAM의 페이지에 요청되는 작은 크기의 덮어 쓰기를 허용함으로써, DRAM과 PRAM 사이의 페이지 이동으로 인한 페이지 크기(4KB)의 PRAM 쓰기 요청을 제거하였다. 또한, 페이지 이동으로 인해 발생하는 불필요한 스토리지의 쓰기 요청을 제거하기 위해 제안 기법은 선택적인 페이지 이동 기법과 최근 참조되지 않은 클린 페이지를 우선 교체하는 기법을 이용한다. 제안 기법의 성능을 평가하기 위해, 시뮬레이션 기반(trace-driven simulation)의 성능 평가를 수행하였으며 시뮬레이션의 결과를 테스트 보드에서 재생(replay)함으로써 실제 PRAM의 성능을 측정하였다. 그 결과, 제안 기법은 전통적인 CLOCK 알고리즘에 비해 최고 92%까지 PRAM의 쓰기 횟수를 감소시키고 PRAM 테스트 보드에서 최대 62%까지 수행시간을 향상시키는 것을 확인하였다.

본 논문의 구성은 다음과 같다. 2장에서는 지금까지

수행된 하이브리드 메모리 구조에서의 버퍼 캐시 정책에 대해서 살펴본다. 3장에서는 본 논문에서 제안하는 기법에 대해서 상세하게 설명하고 4장에서는 제안 기법의 성능을 평가한다. 마지막으로 5장에서 본 논문의 결론을 맺는다.

2. 관련 연구

기존 연구는 DRAM과 PRAM을 모두 사용하는 하이브리드 메모리 구조의 버퍼 캐시 교체 정책으로 집중되었다. 대부분의 하이브리드 메인 메모리 기반의 기존 연구는 DRAM과 PRAM의 장점을 모두 얻기 위해, 읽기가 빈번하게 요청되는 페이지를 DRAM에 유지하고 쓰기가 빈번하게 요청되는 페이지를 PRAM에 유지하는 정책을 이용한다.

PDRAM은 PRAM의 제한된 수명을 고려하여 PRAM의 페이지를 할당하고 특정 PRAM 페이지에 빈번한 쓰기 요청이 발생하는 경우, 해당 페이지의 내용을 DRAM으로 복사시킴으로써 하이브리드 메인 메모리에서의 페이지 이동을 지원한다[4]. 또 다른 버퍼 캐시 교체 정책은 4개의 LRU 큐(queue)을 기반으로 페이지 참조(cache hit)가 발생할 때마다 페이지 이동을 수행함으로써 PRAM 쓰기 횟수를 감소시킨다[5]. CLOCK-DWF의 경우, 쓰기 요청에 의한 새로운 페이지 할당은 DRAM에서 처리하고 읽기 요청에 의한 새로운 페이지 할당은 PRAM에서 처리한다. 그리고 PRAM의 페이지에 쓰기 요청이 발생하는 경우, 해당 페이지를 DRAM으로 복사함으로써 PRAM의 쓰기 횟수를 감소시킨다[6]. 그러나, 지금까지 제안된 기법들은 작은 크기의 쓰기 요청에도 페이지 이동을 위해 페이지 단위(4 KB)의 쓰기 요청을 수행하는 문제점을 가지고 있다. 또한, 기존 연구는 덮어 쓰기가 가능한 경우 발생하지 않는 페이지 교체가 빈번하게 발생하는 문제점을 가지고 있다.

3. 제안 기법

본 논문에서는 DRAM에 비해 느린 PRAM 쓰기 성능과 제한된 수명(endurance) 문제를 완화시키기 위해 DRAM과 PRAM으로 구성된 하이브리드 메인 메모리(hybrid main memory) 구조에서의 버퍼 캐시 교체 정책을 제안한다. 제안 기법은 버퍼 캐시에서의 시간 지역성(temporal locality)을 유지하며, DRAM과 PRAM의 성능 특성을 활용하기 위해 2개의 CLOCK 알고리즘으로 버퍼 캐시의 페이지(page)을 관리한다. 그림 2는 본 논문에서 제안하는 DRAM CLOCK과 PRAM CLOCK의 관계를 보여준다. 제안 기법은 전통적인 CLOCK 알고리즘의 버퍼 캐시 교체 정책을 이용하지만 두 가지의

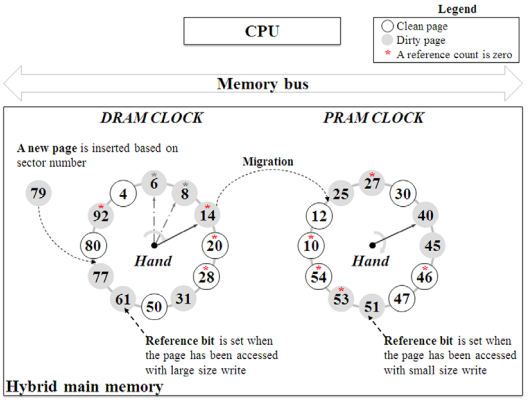

그림 2 제안 기법의 페이지 관리

Fig. 2 Page Management of Proposed Scheme

차이점을 가지고 있다. 첫 번째, 그림 2에서 보여주듯이 버퍼 캐시에서 교체되는 페이지의 공간 지역성(spatial locality)을 높이기 위해 CLOCK 내에서의 페이지를 스토리지의 섹터 번호(sector number)을 기준으로 정렬하여 관리한다. 이는, 교체되는 페이지가 스토리지로 반영될 때, 순차적인 쓰기를 수행함으로써 스토리지의 쓰기 응답 시간(response time)을 감소시킨다[7,8]. 두 번째, 제안 기법은 큰 크기의 쓰기 요청이 빈번한 페이지를 DRAM에 유지시키기 위해 CLOCK의 참조 비트(reference bit) 설정 방법을 개선하였으며 참조 비트 설정에 대한 자세한 설명은 3.2절과 3.3절에서 다룬다.

3.1 DRAM CLOCK의 페이지 관리 정책

DRAM은 PRAM에 비해 쓰기 횟수의 제한이 없기 때문에 CPU에서 요청한 페이지가 하이브리드 메인 메모리에 존재하지 않는 경우(cache miss), 제안 기법은 DRAM에 새로운 페이지를 위치시킨다. 또한, 그림 1에서의 성능 결과를 기반으로 큰 크기의 쓰기 요청이 빈번한 페이지를 DRAM에 오랫동안 유지시키기 위해, 페이지의 참조 비트를 이용한다. 페이지의 참조 비트 초기값은 쓰기 요청에 의해 새롭게 할당된 페이지는 1로 설정하고 읽기 요청에 의해 새롭게 할당된 페이지는 0으로 설정한다. 제안 기법은 큰 크기의 쓰기 요청이 빈번한 페이지와 읽기 요청이 빈번한 페이지를 DRAM에 오랫동안 유지시키기 위해, 페이지에 큰 크기의 쓰기 요청이 발생하거나 읽기 요청이 발생하는 경우(cache hit), 요청된 페이지의 참조 비트를 1로 설정한다. 만약, DRAM에 여유 페이지가 존재하지 않아 CLOCK에서의 페이지 교체(Eviction)가 필요하다면, 전통적인 CLOCK과 동일하게 hand를 이동시키며 참조 비트가 0인 페이지를 검색한다. 만약, 참조 비트가 0인 페이지의 더티 비트(dirty bit)가 1이라면, 해당 페이지는 큰 크기의 쓰기 요청이

```
void DRAM_CLOCK(page *p) {
  if (p is not in the DRAM) {
    if (the DRAM is full) {
      v = select_dramvictim();
      if (PRAM is not full or clean page exists in PRAM)
        migrate v from DRAM to PRAM;
      else
        evict v;
    }
    insert p into the DRAM based on sector number;
  }
  if (p.status is clean or p.updatesize > threshold)
    p.reference_count = 1;
}
```

그림 3 DRAM CLOCK의 핵심 코드
Fig. 3 Pseudo-code of DRAM CLOCK

최근에 발생하지 않은 더티 페이지이기 때문에 해당 페이지를 PRAM으로 이동시킨다. 반면, 참조 비트가 0인 페이지의 더티 비트가 0이라면, 해당 페이지는 교체될 때 스토리지로 쓰기 요청을 발생시키지 않는 클린 페이지(clean page)이므로 PRAM의 페이지 상태에 따라서 선택적으로 페이지의 이동을 수행한다. 만약, PRAM에 여유 페이지가 존재하거나 PRAM CLOCK에 포함된 클린 페이지 중 참조 비트가 0인 페이지가 존재한다면 DRAM에서 교체되는 페이지는 PRAM으로 이동하고 그렇지 않다면 해당 클린 페이지는 DRAM에서 단순 교체된다. 그림 3은 DRAM CLOCK의 페이지 관리 정책의 핵심 코드를 보여준다.

3.2 PRAM CLOCK의 페이지 관리 정책

DRAM CLOCK과 동일하게 PRAM CLOCK도 DRAM에서 이동된 페이지의 더티 비트가 1인 경우, 해당 페이지의 참조 비트를 1로 초기화하고 더티 비트가 0인 경우, 해당 페이지의 참조 비트를 0으로 설정한다. 반면, PRAM CLOCK은 DRAM CLOCK과 반대로 작은 크기의 쓰기 요청이 빈번하게 요청되는 페이지와 읽기 요청이 빈번하게 요청되는 페이지를 PRAM에 유지시키기 위해, PRAM에 위치한 페이지에 작은 크기의 쓰기 요청이 발생하거나 읽기 요청이 발생한 경우, 해당 페이지의 참조 비트를 1로 설정하고 큰 크기의 쓰기 요청이 발생한 경우, 참조 비트를 설정하지 않는다. 만약, PRAM에 여유 페이지가 존재하지 않으면, PRAM CLOCK 역시 CLOCK의 hand을 이용하여 참조 비트가 0인 페이지를 찾고 해당 페이지를 스토리지로 교체시킨다. 그림 4는 PRAM CLOCK의 페이지 관리 정책의 핵심 코드를 보여준다.

```
void PRAM_CLOCK(page *p) {
  if (p is not in the PRAM) {
    if (the PRAM is full) {
      v = select_pramvictim();
      evict v;
    }
    insert p into the PRAM based on sector number;
  }
  if (p.status is clean or p.updatesize < threshold)
    p.reference_count = 1;
}
```

그림 4 PRAM CLOCK의 핵심 코드
Fig. 4 Pseudo-code of PRAM CLOCK

4. 성능 평가

4.1 실험 환경

본 논문에서 제안하는 버퍼 캐시 교체 기법과 전통적인 CLOCK 알고리즘의 성능을 자세하게 평가하기 위해 본 논문에서는 트레이스 기반 시뮬레이션(trace-driven simulation)을 수행하였다. CLOCK 알고리즘은 DRAM과 PRAM 간의 페이지 이동 기능을 고려하고 있지 않기 때문에 DRAM의 용량이 부족한 경우, DRAM에서 가장 오랫동안 접근되지 않은 페이지를 PRAM으로 이동시키도록 DRAM과 PRAM간의 페이지 이동 기능을 구현하였다. 또한, 시뮬레이션의 결과를 실제 PRAM에 재생(replay)함으로써 작은 크기의 쓰기가 PRAM과 시스템 전체 성능에 미치는 영향을 확인하였다. 실험에 사용한 실제 테스트 보드는 ARM 코어를 사용하며 SD card을 스토리지로 사용한다. 또한, 512 MB의 PRAM과 1.25 GB의 DRAM을 보드의 메모리 버스(memory bus)로 연결함으로써 바이트 단위 접근이 가능하도록 하였다[9].

본 논문에서는 다양한 환경에서의 성능 측정을 위해 시뮬레이션의 입력으로 모바일 트레이스와 서버 트레이스를 모두 사용하였으며 표 1은 각 트레이스의 특징을 보여준다. 모바일 트레이스는 실제 모바일 디바이스인 Google Nexus 7 태블릿(Tablet)에서 수집되었다[10]. 수집한 Web 워크로드는 Web 어플리케이션을 수행하는 동안에 수집한 워크로드이며, Mixed 워크로드는 YouTube, Gallery, Facebook, 그리고 Maps 등의 다양한 어플리케이션을 수행할 때 수집한 워크로드이다. 실험 환경의 확장을 위한 서버 트레이스는 가장 널리 사용되고 있는 Filebench 벤치마크를 사용하였으며 Webserver을 수행하는 동안에 수집되었다[11]. 또한, 본 논문에서는 버퍼 캐시의 용량에 따른 알고리즘의 성능 변화를 확인하기

표 1 메모리 트레이스 특성
Table 1 Characteristics of memory trace

	메모리 사용량	평균 쓰기 요청 크기
web	20,428 KB	1599 Byte
mixed	57,320 KB	1625 Byte
webserver	39,540 KB	3035 Byte

위해 워크로드의 최대 메모리 사용량(memory footprint)을 측정하였으며 모든 실험은 워크로드의 최대 메모리 사용량 대비 10%부터 100%까지 캐시 용량을 증가시키며 수행되었다. 마지막으로, PRAM은 DRAM에 비해 대략 3배 정도 밀도(Density)가 높다고 예측되기 때문에 각 실험에서의 DRAM과 PRAM의 비율은 1:3으로 고정하였다[5]. 즉, 버퍼 캐시 크기가 100%인 경우, 워크로드가 사용하는 모든 페이지가 버퍼 캐시에 존재하기 때문에 페이지 교체(eviction)가 발생하지 않으며 버퍼 캐시 내부의 각 메모리양은 DRAM 25%, PRAM 75%로 구성된다.

4.2 실험 결과

본 논문에서 제안하는 알고리즘의 시간 지역성(temporal locality)을 확인하기 위해, 메모리 캐시 적중률(cache hit ratio)을 평가하였다. 그림 3은 캐시 용량이 최대 메모리 사용량에 비해 10%부터 100%까지의 각 트레이스 별 캐시 적중률을 보여준다. 그림 5에서 보여주듯이, 제안 기법은 DRAM의 클린 페이지를 PRAM으로 선택적으로 이동시킴에도 불구하고 전통적인 CLOCK 알고리즘과 비슷한 캐시 적중률을 보여준다. 특히, 임의 읽기 요청이 많은 Webserver 워크로드(Random Read-intensive Workload)의 경우, 캐시 용량이 50%보다 작을 때 기존 CLOCK보다 높은 캐시 적중률을 보여준다. 이러한 결과는 DRAM CLOCK과 PRAM CLOCK에서 빈번하게 접근되는 클린 페이지를 오랫동안 유지시키기 위해 참조 비트를 설정하는 반면 쓰기 요청이 발생하는 페이지는 선택적으로 참조 비트를 설정하기 때문이다.

PRAM은 DRAM에 비해 제한된 수명을 가지고 있기 때문에 PRAM의 쓰기 횟수는 하이브리드 메인 메모리의 수명을 평가하는 중요한 평가 요소이다. 이에, 본 논문에서는 제안 기법과 CLOCK 알고리즘의 PRAM 쓰기 횟수를 비교하였다. 그림 6은 제안 기법의 PRAM 쓰기 횟수를 CLOCK의 쓰기 횟수에 정규화하여 보여준다. 실험 결과, 제안 기법의 PRAM 쓰기 횟수가 Web 워크로드의 경우, CLOCK에 비해 평균 11.4%, 최대 27% 감소하는 것을 확인하였으며 Mixed 워크로드의 경우, 평균 10.8%, 최대 40.7% 감소하는 것을 확인하였다. 특히, 서버 워크로드인 Webserver의 경우, CLOCK에 비해 평균 42.6%, 최대 92.3%의 PRAM 쓰기 횟수가 감소하는 것을 확인하였다. 이러한 PRAM 쓰기 횟수의 감소는 빈번하게 접근되지 않는 DRAM CLOCK의 클린 페이지를 PRAM CLOCK으로 이동시키지 않고 단순 교체함으로써 가능하다. 그러나 Mixed 워크로드의 경우, 버퍼 캐시 용량이 작은 일부 구간에서 CLOCK

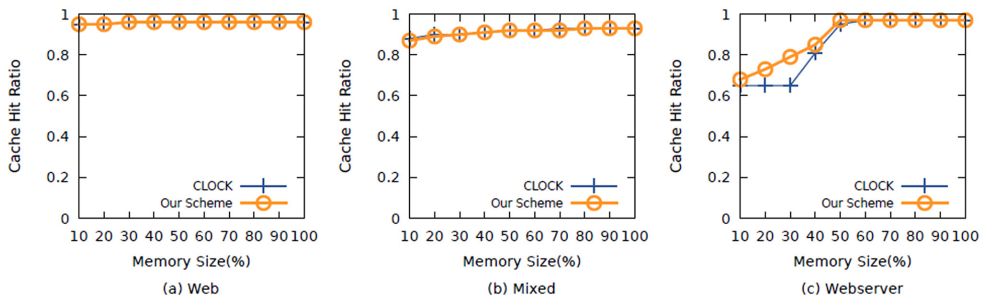

그림 5 캐시 적중률
Fig. 5 Cache Hit Ratio

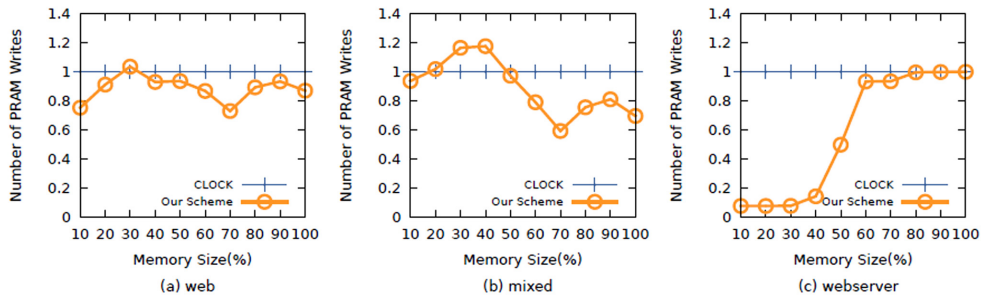

그림 6 PRAM 쓰기 횟수
Fig. 6 The Number of PRAM Writes

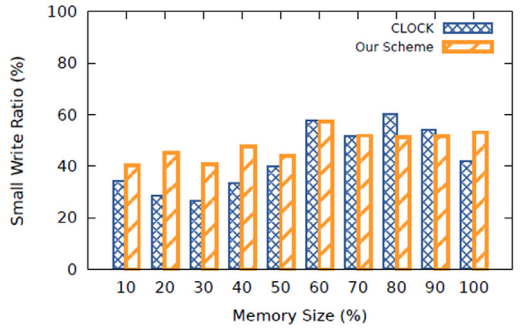

그림 7 PRAM의 작은 쓰기 요청 비율
Fig. 7 Small Write Ratio of PRAM

보다 쓰기 횟수가 증가하는 것을 확인할 수 있다. 해당 구간의 쓰기 횟수가 실제 PRAM의 쓰기 성능에 어떠한 영향을 미치는지 자세하게 확인하기 위해 Mixed 워크로드의 시뮬레이션 결과를 분석하였다. 그림 7은 PRAM의 쓰기 횟수 중에서 512 바이트 이하의 작은 쓰기가 발생한 쓰기 비율을 보여주며 그림 4에서 쓰기 횟수가 증가한 구간에 작은 크기의 쓰기가 빈번하게 발생한 것을 확인할 수 있다. 이는, 제안 기법이 작은 크기의 쓰기 요청이 빈번한 페이지를 PRAM에 오랫동안 유지시키기 때문이다.

PRAM의 쓰기 횟수와 작은 크기의 PRAM 쓰기가 전체 PRAM의 쓰기 요청 양과 어떠한 관계가 있는지 확인하기 위해 전체 PRAM의 쓰기 요청 양을 확인하였다. 그림 8은 제안 기법이 PRAM에 요청한 전체 쓰기 양을 CLOCK이 PRAM에 요청한 전체 쓰기 양에 정규화하여 보여준다. 실험 결과, 모든 워크로드와 버퍼 캐시 용량에서 제안 기법이 CLOCK보다 적은 양의 쓰기를 PRAM에 요청한다는 사실을 확인할 수 있다. 이러한 실험 결과는 그림 5에서 확인하였듯이, 작은 크기의 쓰기 요청이 빈번한 페이지를 PRAM에 오랫동안 유지시키고 PRAM의 페이지에 바이트 단위의 덮어 쓰기(in-place update)을 수행하기 때문이다.

마지막으로, 본 논문에서는 DRAM과 PRAM의 바이트 단위의 읽기 및 쓰기와 버퍼 캐시 교체로 인한 스토리지 I/O가 시스템의 전체 성능에 어떠한 영향을 미치는지 확인하기 위해 제안 기법과 CLOCK의 시뮬레이션 결과를 실제 PRAM 테스트 보드에서 재생하였다. 그림 9는 각 알고리즘을 재생하였을 때 전체 소모 시간을 보여준다. 보드 테스트 결과, 제안 기법이 CLOCK에 비해 모바일 워크로드에서는 최대 12.6% 서버 워크로드에서는 최대 62%까지 소모시간을 단축시키는 것을 확인하

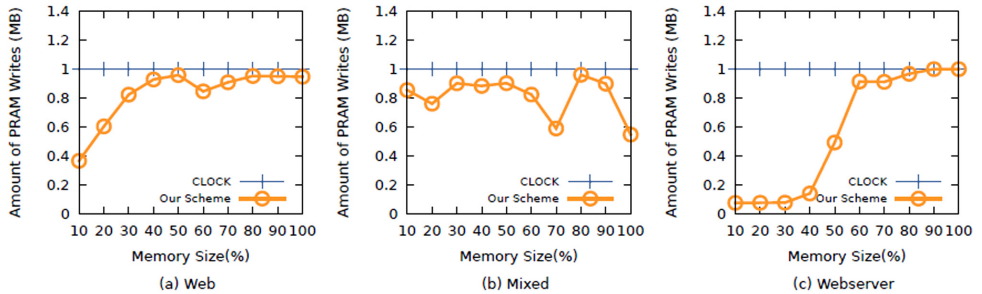

그림 8 PRAM의 전체 쓰기 요청 양
Fig. 8 The Amount of PRAM Writes

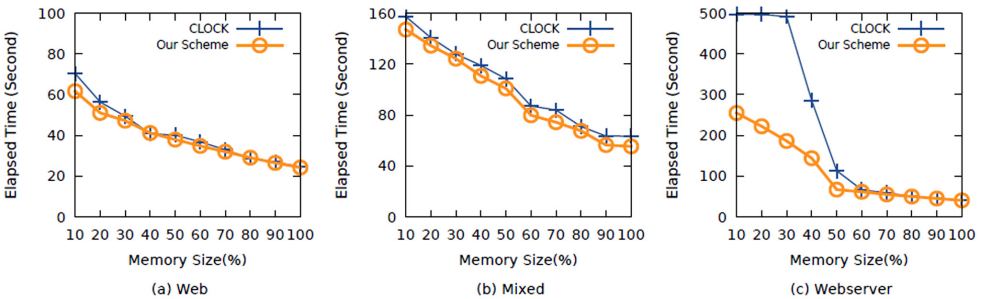

그림 9 실제 보드 상에서의 전체 성능 평가
Fig. 9 Total Elapsed Time on Real Test Board

였다. 또한, 보드에서의 결과와 시뮬레이션의 결과를 전체적으로 분석함으로써 작은 크기의 쓰기 요청을 PRAM에서 수행하고 선택적인 이동을 통해 PRAM의 쓰기 횟수를 줄이는 것이 성능에 많은 영향을 미친다는 사실을 확인하였다. 특히, 작은 크기의 버퍼 캐시 용량으로 인해 페이지 교체가 많은 경우, 제안하는 기법의 순차 쓰기가 CLOCK보다 스토리지의 성능을 향상시킬 수 있음을 확인하였다.

4. 결 론

본 논문에서는, DRAM과 PRAM으로 구성된 하이브리드 메인 메모리 환경에서, PRAM의 쓰기 요청 양에 따른 성능 차이를 고려하는 새로운 버퍼 캐시 교체 정책을 제안하였다. 제안 기법은 작은 크기의 쓰기 요청이 빈번한 페이지를 PRAM에 위치시키고 작은 크기의 요청에 대한 덮어 쓰기를 허용함으로써 PRAM의 바이트 단위 쓰기 성능 특성을 최대화하였다. 그 결과 시스템의 전체 성능을 향상 시킬 수 있었다. 또한, 선택적 이동을 수행함으로써 PRAM의 쓰기 횟수 및 불필요한 스토리지 쓰기 요청을 상당히 감소시켰다. 선택적 이동은 클린 페이지의 캐시 적중률을 감소시킬 수 있으나 낸드 기반의 SD card는 읽기 속도가 쓰기 속도보다 빠르기 때문에 전체성능에 영향을 미치지 않는다. 실험 결과, 전통적인 CLOCK 알고리즘에 비해 제안 기법은 PRAM의 쓰기 횟수는 최고 92%까지 감소시킨다. 또한, 제안 기법은 실제 PRAM 테스트 보드에서 CLOCK보다 최대 62%까지 수행시간을 향상시킨다.

References

[1] H. Kim, S. Seshadri, C. L. Dickey, and L. Chiu, "Evaluating Phase Change Memory for Enterprise Storage Systems: A Study of Caching and Tiering Approaches," *Proc. of the 13th USENIX Conference on File and Storage Technologies*, 2014.

[2] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," *Proc. of the 36th ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2009.

[3] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable High Performance Main Memory System Using Phase-Change Memory Technology," *Proc. of the 36th ACM/IEEE International Symposium on Computer Architecture*, 2009.

[4] G. Dhiman, R. Ayoub, and T. Rosing, "PDRAM: A Hybrid PRAM and DRAM Main Memory System," *Proc. of the 46th ACM/IEEE Design Automation Conference*, 2009.

[5] H. Seok, Y. Park, K.-W. Park, and K. H. Park, "Efficient Page Caching Algorithm with Prediction and Migration for a Hybrid Main Memory," *ACM SIGAPP Applied Computing Review*, Vol. 11, No. 4, pp. 38-48, 2011.

[6] S. Lee, H. Bahn, and S. Noh, "CLOCK-DWF: A Write-History-Aware Page Replacement Algorithm for Hybrid PCM and DRAM Memory Architectures," *IEEE Transactions on Computers*, Vol. 63, No. 9, pp. 2187-2200, 2013.

[7] H. Kim, M. Ryu, and U. Ramachandran, "What is a Good Buffer Cache Replacement Scheme for Mobile Flash Storage?," *Proc. of the 12th ACM SIGMETRICS/PERFORMANCE joint International Conference on Measurement and Modeling of Computer Systems*, 2012.

[8] D. H. Kang, C. Min, and Y. I. Eom, "An Efficient Buffer Replacement Algorithm for NAND Flash Storage Devices," *Proc. of the IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*, 2014.

[9] T. Lee, H. Park, D. Kim, S. Yoo, and S. Lee, "FPGA-based prototyping systems for emerging memory technologies," *Proc. of the 25th IEEE International Symposium on Rapid System Prototyping*, 2014.

[10] Android Open Source Project [Online]. Available: http://source.android.com/index.html

[11] Filebench [Online]. Available: http://www.solarisinternals.com/wiki/index.php/FileBench

강 동 현
정보과학회논문지
제 42 권 제 1 호 참조

엄 영 익
정보과학회논문지
제 42 권 제 1 호 참조