ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 42, No. 11, pp. 1314-1321, 2015. 11
http://dx.doi.org/10.5626/JOK.2015.42.11.1314

모바일 애플리케이션의 특성을 이용한 하이브리드 메모리 기반 버퍼 캐시 정책

(Hybrid Main Memory based Buffer Cache Scheme by Using Characteristics of Mobile Applications)

오 찬 수 † (Chansoo Oh)　강 동 현 † (Dong Hyun Kang)　이 민 호 † (Minho Lee)　엄 영 익 †† (Young Ik Eom)

요 약 모바일 디바이스는 데스크톱이나 서버 등 일반 컴퓨터 시스템과 마찬가지로 주기억장치와 스토리지와의 성능 차이를 완화시키기 위해 버퍼 캐시를 사용한다. 그러나 DRAM 은 저장된 데이터를 유지하기 위해 주기적인 refresh 연산을 수행함으로써 제한된 크기의 배터리 소모를 가속화하는 문제점을 가지고 있다. 본 논문에서는 모바일 디바이스 환경에서 배터리의 수명을 연장하기 위해 DRAM과 비휘발성 메모리인 PCM으로 구성된 하이브리드 메인 메모리 구조기반의 버퍼캐시 정책을 소개한다. 또한, PCM의 성능 및 내구성 특성을 최적화시키기 위해 프로세스 상태 기반의 새로운 버퍼 캐시 정책을 제안한다. 제안 기법은 포그라운드 및 백그라운드 애플리케이션이 사용하는 페이지를 서로 다른 방법으로 배치함으로써 소량의 DRAM으로도 포그라운드 애플리케이션의 빠른 응답성을 보장한다. 실험 결과, 제안 기법은 포그라운드 애플리케이션의 총 수행시간을 평균 58% 감소시켰으며 전력 소비량도 평균 23% 감소시키는 것을 확인하였다.

키워드: 하이브리드 메모리, 버퍼 캐시 정책, 포그라운드 애플리케이션, phase change memory

Abstract Mobile devices employ buffer cache mechanisms, just as in computer systems such as desktops or servers, to mitigate the performance gap between main memory and secondary storage. However, DRAM has a problem in that it accelerates battery consumption by performing refresh operations periodically to maintain the stored data. In this paper, we propose a novel buffer cache scheme to increase the battery lifecycle in mobile devices based on a hybrid main memory architecture consisting of DRAM and non-volatile PCM. We also suggest a new buffer cache policy that allocates buffers based on process states to optimize the performance and endurance of PCM. In particular, our algorithm allocates each page to the appropriate position corresponding to the state of the application that owns the page, and tries to ensure a rapid response of foreground applications even with a small amount of DRAM memory. The experimental results indicate that the proposed scheme reduces the elapsed time of foreground applications by 58% on average and power consumption by 23% on average without negatively impacting the performance of background applications.

Keywords: hybrid memory system, buffer cache scheme, foreground application, phase change memory

· 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학 ICT연구센터육성지원사업의 연구결과로 수행되었음(IITP-2015-(H8501-15-1015))

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

논문접수 : 2015년 5월 19일
(Received 19 May 2015)
논문수정 : 2015년 8월 26일
(Revised 26 August 2015)
심사완료 : 2015년 8월 26일
(Accepted 26 August 2015)

정보과학회논문지 제42권 제11호(2015. 11)

1. 서 론

최근 멀티태스킹 환경의 고성능 멀티코어 모바일 디바이스가 대중화 되면서 빠른 배터리 소모를 극복하기 위한 다양한 연구가 활발히 진행되고 있다[1-4]. 특히, 대부분의 모바일 디바이스가 버퍼 캐시로 사용하고 있는 DRAM은 저장된 데이터를 유지하기 위해 주기적인 전력 소모를 필요로 하며, 이는 전체 시스템의 약 40%에 이른다[5]. 이러한 DRAM의 휘발성 메모리의 특징을 완화하기 위한 대안으로 PCM(Phase Change Memory)과 같은 비휘발성 메모리에 대한 연구들이 활발히 진행되고 있다[5-8]. PCM은 비휘발성 메모리이기 때문에 데이터 유지를 위해 주기적인 전력소모를 필요로 하지 않으며 바이트 단위로 접근이 가능하다는 장점을 가지고 있다. 하지만 PCM은 DRAM에 비해 약 1~2배 느린 읽기 속도와 약 10배 느린 쓰기 속도의 특징을 가지고 있으며 전체 쓰기 횟수(Endurance)가 10^7~10^8회로 DRAM보다 제한되는 특징을 가지고 있다.

이러한 PCM의 특징을 고려하기 위해 최근 DRAM과 PCM을 함께 사용하는 하이브리드 메인 메모리 구조의 버퍼 캐시 정책이 제안되었다[5-7]. 그러나 지금까지의 연구는 주로 데스크톱 및 서버 환경을 기반으로 메모리 시스템의 성능에 중점을 두어 이루어 졌기 때문에 제한된 크기의 배터리를 가지고 있는 스마트폰 환경에는 적합하지 않다. 따라서 본 논문에서는 스마트폰의 전력 소모량을 줄이고자 소량의 DRAM과 PCM을 함께 사용하는 하이브리드 메인 메모리 기반의 버퍼 캐시 기법을 제안한다. 특히, 제안 기법은 DRAM보다 느린 PCM의 읽기 및 쓰기 특징을 고려하기 위해 포그라운드 애플리케이션의 페이지들을 읽기 및 쓰기 속도가 빠른 DRAM에 배치시키며, 백그라운드 애플리케이션 페이지들은 DRAM보다 상대적으로 읽기 및 쓰기 속도가 느린 PCM상에 배치시킨다. 이는 백그라운드 애플리케이션이 포그라운드 애플리케이션보다 상대적으로 적은 횟수의 읽기 및 쓰기를 요청하며, 사용자에게 직접적인 영향을 주지 않기 때문에 사용자는 그 성능 차이를 느끼기 어렵다는 것을 기반으로 한다. 실험 결과, 본 논문에서 제안하는 기법은 기존의 버퍼 캐시 기법들에 비해 포그라운드 애플리케이션의 수행 시간을 평균 53% 단축시켰다. 또한, 포그라운드 애플리케이션의 페이지들이 PCM 또는 보조 저장장치(예, eMMC, MicroSD Card)로 요청하는 쓰기 횟수를 줄여 전체 모바일 디바이스에서 소모되는 에너지를 크게 줄일 수 있었다.

본 논문의 구성은 다음과 같다. 제2장에서 PCM과 안드로이드 플랫폼에 대한 관련 연구를 살펴보고, 제3장에서 제안하는 기법의 특성과 동작 방법에 대해 기술한다. 제4장에서는 실험을 통하여 제안하는 기법을 기존의 버퍼 캐시 기법과 비교 평가하며, 마지막으로 제5장에서 결론 및 향후 연구 과제를 서술한다.

2. 관련연구

본 장에서는 비휘발성 메모리인 PCM의 특징에 대해 알아보고 포그라운드 애플리케이션과 백그라운드 애플리케이션 구분을 위해 안드로이드 플랫폼에서 사용하고 있는 중요성 계층(Importance Hierarchy)에 대해 설명한다.

2.1 PCM

PCM은 DRAM을 대체하기 위한 비휘발성 메모리 기술 중 하나이며 최근 PCM을 활용한 연구가 활발히 진행되고 있다[5-7]. PCM은 표 1에서 보는 것과 같이 비휘발성의 특징을 가지고 있어 저장된 데이터를 유지하기 위한 주기적인 전력소모가 필요없다[8]. 하지만 DRAM에 비해 상대적으로 느린 읽기 및 쓰기 속도를 가지며 총 쓰기 횟수가 최대 10^8번으로 제한되는 특징도 가지고 있다. 따라서 하이브리드 메인 메모리 구조에 대한 기존 연구들은 PCM의 제한적인 쓰기 횟수와 느린 속도를 고려하여 PCM에 페이지를 쓰는 횟수를 줄이고자 하였다. 그러나, 최근 제안된 DRAM과 PCM을 함께 사용하는 하이브리드 메인 메모리 구조 연구들은 데스크톱 및 서버 환경에 최적화 되어있기 때문에 제한된 크기의 배터리를 사용하는 모바일 디바이스에는 적합하지 않다[5-7].

표 1 PCM과 DRAM의 특성
Table 1 Characteristic of DRAM and PCM

	DRAM	PCM
Non-volatile	No	Yes
Read Latency	50ns	50~100ns
Write Latency	~20-50ns	~1us
Read Energy	~0.1nJ/b	~0.1nJ/b
Write Energy	~0.1nJ/b	~1nJ/b
Static Energy	~W/GB	<<0.1W
Endurance	∞	108

2.2 안드로이드 플랫폼

안드로이드 플랫폼은 모바일 디바이스(예, 스마트폰, 태블릿) 환경에서 가장 널리 사용되고 있는 소프트웨어 플랫폼 중 하나이다[9]. 안드로이드 플랫폼은 모바일 디바이스에 최적화된 커널을 제공하기 위해 리눅스 커널 메인 라인으로부터 Wakelocks[10], Binder[11], Low-memorykiller[12] 등과 같은 기능들을 도입하고 있다. 하지만, 제한된 크기와 가격 경쟁력을 이유로 모바일 디바이스에서는 CPU, 메모리, 배터리와 같은 하드웨어 자

원이 물리적으로 제한되며 다수의 애플리케이션이 동시에 수행될 가능성이 높다. 이때 다수의 백그라운드 애플리케이션으로부터 포그라운드 애플리케이션의 성능을 보장하기 위해 CPU나 메모리 등의 자원할당과 프로세스간 스케줄링 방법에 대한 연구들이 진행되고 있다[13]. 또한 한정된 크기의 배터리 자원의 사용 시간을 늘리기 위하여 다양한 연구가 진행되고 있다[3,4,14].

안드로이드 플랫폼은 백그라운드로 전환된 다수의 애플리케이션들을 메모리상에 가능한 오래 유지시키려 한다. 하지만 새롭게 실행되는 프로세스나 다른 중요한 프로세스의 동작을 위해 메모리가 부족하게 될 경우 백그라운드 프로세스를 제거하여 필요한 메모리를 확보한다. 이때 제거할 프로세스를 선택하는 우선순위를 주기 위해 안드로이드 플랫폼에서는 중요성 계층을 만들어 프로세스를 분류하며[15], 중요성 계층에 따른 프로세스 분류는 그림 1과 같다.

1) 포그라운드 프로세스(Foreground Process): 유저가 현재 사용하고 있는 프로세스를 뜻하며 일반적인 스마트폰 사용 환경에서는 극히 적은 수의 프로세스만 포그라운드 프로세스로 분류된다[15].
2) 비저블 프로세스(Visible Process): 사용자가 조작하는 포그라운드 프로세스는 아니지만 현재 화면에 보여지는 프로세스이다. 매우 중요한 프로세스로 취급되며 포그라운드 프로세스가 동작하고 있는 동안 제거되지 않는다[15].
3) 서비스 프로세스(Service Process): 현재 유저가 수행하고 있는 애플리케이션 동작에 직접적인 영향은 주지 않지만 음악 파일을 재생한다거나 네트워크 데이터 전송 등의 일을 한다. 서비스 프로세스들은 포그라운드나 비저블 프로세스를 동작시키기에 하드웨어 자원이 부족하지 않는 한 제거되지 않는다[15].
4) 백그라운드 프로세스(Background Process): 사용자에게 어떠한 직접적인 영향도 주지 않는 상태의 프로세스이다. 포그라운드, 비저블, 서비스 프로세스의 정상적인 동작 상태를 유지하기 위해 언제든지 제거될 수 있으며 시스템 메모리가 부족할 경우 가장 최근에 사용된 프로세스가 제일 마지막에 제거될 수 있도록 LRU(Least Recently Used) 리스트로 관리된다[15].
5) 엠프티 프로세스(Empty Process): 활성화된 애플리케이션 컴포넌트를 가지고 있지 않고 비어있는 프로세스이다. 새로운 애플리케이션이 실행될 때 스타트업 타임을 줄이기 위한 캐싱의 목적으로 사용되며 필요한 경우 제거될 수 있다[15].

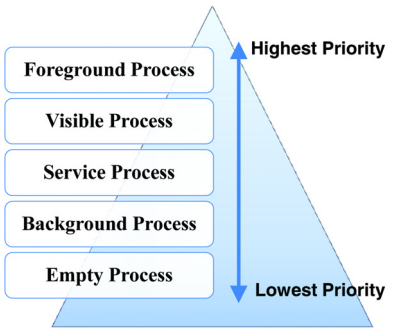

그림 1 중요성 계층
Fig. 1 Importance hierarchy

본 논문에서는 이러한 애플리케이션 중요성 계층을 근거로 하여 애플리케이션의 상태에 따라 데이터 저장 위치를 다르게 배치하는 하이브리드 메인 메모리 기반의 버퍼 캐시 정책을 제안한다.

3. 제안기법

본 논문에서는 모바일 디바이스의 전력 소모를 줄이기 위해 DRAM과 PCM으로 구성된 하이브리드 메인 메모리 구조 기반의 버퍼 캐시 정책을 소개한다. 또한, 사용자에게 직접적인 영향을 끼치지 않는 프로세스들을 백그라운드 프로세스로 분류하고 해당 프로세스가 사용하는 페이지를 PCM상에서 위치시키도록 함으로써 PCM의 특성을 고려하고 포그라운드 프로세스의 빠른 응답성을 보장한다.

단일 디스플레이로 구성되어 있는 일반적인 스마트폰은 하나의 포그라운드 애플리케이션과 다수의 백그라운드 애플리케이션이 동시에 수행될 수 있다. 이때 백그라운드에서 동작하는 애플리케이션들은 많은 읽기 및 쓰기 요청을 발생시키지 않으며 대부분의 읽기 및 쓰기 요청은 애플리케이션의 푸쉬(Push) 기능(예, 미디어파일의 재생, 메신저 수신)을 제공하기 위한 요청이다. 특히, 백그라운드 애플리케이션은 사용자에게 직접적인 영향을 미치지 않기 때문에 백그라운드 애플리케이션의 읽기 및 쓰기 속도는 포그라운드 애플리케이션에 비해 중요하지 않다.

이에, 본 논문에서 제안하는 기법은 페이지의 시간적 지역성(Temporal Locality)을 활용하기 위해 전통적인 LRU 기법을 기반으로 하였으며 사용자의 빠른 응답성을 요구하는 포그라운드 애플리케이션의 데이터를 저장하고 있는 페이지는 DRAM에 위치시키고 백그라운드 애플리케이션의 데이터를 저장하고 있는 페이지는 PCM에 위치시키도록 확장하였다. 또한, 제안 기법은 DRAM과 PCM간의 페이지 이동(Migration)에 따른 오버헤드를 줄이기 위하여 LRU 기법에 참조 카운트(Reference Count)을 추가로 이용한다.

포그라운드 애플리케이션이 사용하는 페이지를 DRAM

에 위치시키기 위해, 제안 기법은 포그라운드 애플리케이션이 요청한 페이지가 메인 메모리에 없을 경우(Cache Miss), 새로운 페이지를 DRAM의 MRU(Most Recently Used)에 우선적으로 할당한다. 반면, 백그라운드 애플리케이션이 요청하는 페이지가 메인 메모리에 없을 경우, 새로운 페이지를 PCM의 MRU에 우선적으로 할당한다. 또한, 백그라운드 애플리케이션에서 포그라운드 애플리케이션으로 애플리케이션의 상태가 전환되는 경우, 참조 카운트를 이용하여PCM에 위치한 페이지를 DRAM의 MRU로 이동시킴으로써 페이지 이동에 따른 오버헤드를 감소시킨다.

DRAM과 PCM의 페이지들은 각각 참조 카운트를 가지고 있다. 각 페이지의 참조 카운트는 페이지가 재 참조되면 1씩 증가하고 LRU의 추출후보(Eviction Candidate)가 되면 1씩 감소한다. 각 메모리의 LRU 위치에 있는 페이지의 참조 카운트가 0이면 희생 페이지(Victim)로 선택된다. DRAM에서 선택된 희생 페이지는 PCM의 MRU로 이동되며, PCM에서 선택된 희생 페이지는 보조저장장치로 추출(Eviction)된다.

그림 2는 제안 기법의 페이지 이동 및 추출의 예를 보여주며 그림 3은 핵심 코드를 나타낸다. 그림 2와 그림 3에서 나타냈듯이 새롭게 메모리로 올라오는 페이지들은 메모리를 요청한 애플리케이션의 분류에 따라 각각 DRAM과 PCM의 MRU위치로 쓰여지게 된다. 만약, 백그라운드 애플리케이션에서 포그라운드 애플리케이션으로 전환된 후 PCM에 위치한 페이지가 2번 이상 쓰기 참조가 발생하면 해당 페이지는 빈번하게 접근될 확률이 높다고 판단하여 해당 페이지를 DRAM의 MRU로 이동시킨다(그림 2(a)). 반면, 포그라운드 애플리케이션이 사용하는 페이지일지라도 빈번하게 접근 쓰기 요청이 발생하지 않는 페이지의 경우, 해당 페이지는 이동에 따른 오버헤드를 고려하여 PCM의 MRU 위치로 이동시킨다(그림 2(b)). 이는 쓰기 참조가 빈번하게 발생하는 포그라운드 애플리케이션 페이지들이 느린 쓰기 속도를 가지고 있는 PCM으로 이동됨으로써 모바일 디바이스의 성능이 저하되지 않도록 하기 위함이다. 반면 PCM의 읽기 속도는 DRAM에 비해 크게 느리지 않으므로 PCM에 있는 백그라운드 애플리케이션 페이지들과 포그라운드 애플리케이션의 읽기 전용 페이지들은 재 참조가 발생하여도 DRAM으로 이동되지 않고 PCM의 MRU로 이동한다(그림 3 line 8, line 19). 만약, PCM에서 DRAM으로 페이지를 이동시킬 때 DRAM에 페이지의 여유 공간이 존재하지 않는다면 제안 기법은 DRAM에서 희생 페이지를 선택하여 PCM으로 이동시킨다(그림 2(c)). 또한, PCM에 페이지의 여유 공간이 존재하지 않는다면, 제안 기법은 PCM에서 희생 페이지를 선택하여 보조저장장치로 이동시킨다(그림 2(d)). 또한 포그라운드 애플리케이션의 상태가 백그라운드로 전환되었을

```
1  if page hit occurs then
2      if page is on DRAM then
3          the page is moved to MRU on DRAM;
4      else if page is on PCM then
5          if foreground page && Dirty Page && ref cnt >= 2
           then
6              the page is migrated to MRU of DRAM;
7          else
8              the page is moved to MRU of PCM;
9          continue;
10 if page miss occurs then
11     if page belongs to foreground application then
12         if DRAM is full then
13             if PCM is full then
14                 evict LRU page on PCM to storage;
15             evict LRU page on DRAM to PCM;
16         write a new page on DRAM;
17     else
18         if PCM is full then
19             evict LRU page on PCM to storage;
20         write a new page on PCM;
```

그림 3 제안 기법의 핵심 코드
Fig. 3 Pseudo code

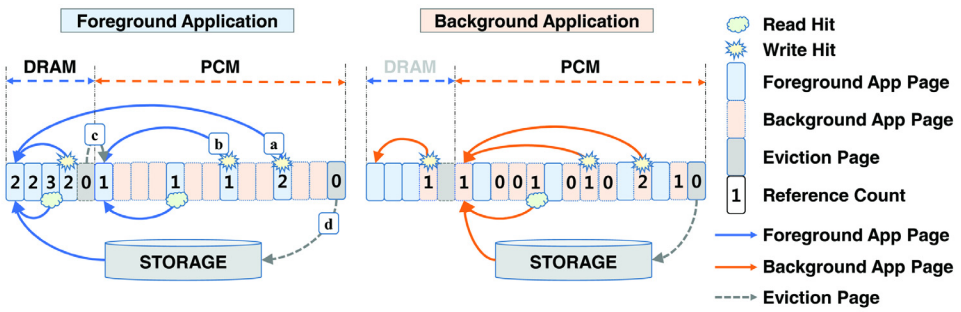

그림 2 제안한 애플리케이션 상태에 따른 버퍼 캐시 기법의 예
Fig. 2 The example of proposed buffer caching algorithm with application state

경우 DRAM에 백그라운드 애플리케이션의 페이지들이 남아있게 된다. 이때 DRAM에서 백그라운드 애플리케이션의 페이지가 재 참조되면 해당 페이지는 참조가 빈번한 페이지로 판단하여 DRAM의 MRU위치로 이동시킨다(그림 3 line 3). 하지만 DRAM에서 재 참조가 발생하지 않은 페이지들은 LRU 정책에 의해 자연스럽게 PCM으로 이동하게 된다.

4. 실험 결과 및 분석

본 논문에서 제안하는 하이브리드 메모리 기반의 버퍼 캐시 정책의 성능을 측정하기 위하여 커널 버전 3.4.0 기반의 안드로이드 킷캣 버전 4.2.2를 운영체제로 쓰는 Google Nexus7를 사용하였다. 실험을 위해 Nexus7의 안드로이드 커널을 수정하여 애플리케이션이 버퍼로 발생시키는 페이지 요청에 대한 트레이스를 수집하였다. 실험을 진행할 당시에 모바일 워크로드 분석을 위한 벤치마크 툴이 존재하지 않아 실제 모바일 디바이스에서 수집한 트레이스를 이용하여 시뮬레이션을 통해 실험을 진행하였다. 또한 다양한 환경의 워크로드를 얻기 위해 여러 방법으로 트레이스를 수집한 결과 백그라운드 상태의 애플리케이션은 그 수가 증가해도 페이지 요청을 거의 발생시키지 않는 것을 확인할 수 있었다. 따라서 백그라운드 애플리케이션의 페이지 요청 횟수를 증가시키기 위하여 인위적으로 메신저 애플리케이션을 통하여 푸쉬 서비스를 발생시켜 백그라운드 애플리케이션의 페이지 요청을 발생시켰다. 이때 페이지를 요청한 애플리케이션의 포그라운드와 백그라운드 상태를 구분하기 위해 커널에서 사용하고 있는 oom_adj값을 참조하였다. oom_adj값은 시스템의 메모리가 부족한 상황에서 종료시킬 프로세스를 선택할 때 참조하는 우선순위 값으로써 일반적인 경우 0부터 15까지 범위를 갖는다. 포그라운드 프로세스는 가장 높은 우선순위인 0값을 가지며 그 외의 프로세스는 init.rc에 정의되어 있는 대로 1부터 15까지 값을 갖는다. 안드로이드 플랫폼에서는 이렇게 리눅스 커널에서 세분화 되어 있는 oom_adj 값 중 다섯 가지를 선택하여 중요성 계층으로 사용하고 있다. 이때 제안하는 기법에서는 안드로이드 중요성계층에서 포그라운드 프로세스로 분류되는 애플리케이션, 즉 oom_adj값이 0인 프로세스들을 포그라운드 애플리케이션으로 구분하여 사용하고 나머지 네 가지 종류의 프로세스들을 백그라운드 어플리케이션으로 구분하여 실험을 진행하였다. 성능 비교를 위하여 LRU 알고리즘과 2QLRU [16] 알고리즘에 대해 동일한 실험을 수행하여 결과값을 도표에 비교하여 나타내었다.

실험에 사용된 첫 번째 트레이스는 웹 브라우저인 Chrome을 포그라운드 애플리케이션으로 사용하였고 백그라운드에서 Hangout(메신저)과 Gmail을 통하여 푸쉬를 발생시켜 수집하였으며 포그라운드와 백그라운드의 페이지 요청 비율은 약 15:1로 발생하였다. 두 번째 트레이스는 다수의 애플리케이션을 포그라운드와 백그라운드로 전환시켜가며 수집하였고 약 3:1의 비율로 포그라운드와 백그라운드 페이지 요청이 발생하였다. 이렇게 수집된 트레이스를 제안하는 기법의 입력으로 받아 수행시킨 결과를 그림 4와 그림 5에 나타내었다. 그래프의 x축은 실험에서 사용한 메모리의 크기로써 각 워크로드에서 필요로 하는 메모리 크기의 10%에서 100%까지를 나타낸다. 그래프의 y축은 전체 워크로드에 대해 본 논문이 제안하는 기법을 사용해 측정한 메모리 적중률 및 포그라운드 애플리케이션의 성능을 비교한 것이다. 실험에서는 전력 소모 감소량을 최대화하기 위하여 DRAM의 비율을 전체 메모리의 5%로 고정하여 실험을 진행하였다. 각 그림에서 (a)는 전체 워크로드에 대한 메모리 적중률을 보여주며 (b)는 포그라운드 애플리케이션의 메모리 적중률, (c)는 LRU 기법을 기준으로 정규화된 포그라운드 애플리케이션의 수행시간을 나타낸다. 시스템 전체의 메모리 적중률과 포그라운드 애플리케이션의 메모리 적중률은 기존의 LRU나 2QLRU 기법과 비교하여 낮은 구간 없이 메모리 사이즈가 커짐에 따라 메모리 적중률도 높아지는 결과를 나타내었다. 그림 5에서 LRU기법과 2QLRU 기법의 그래프가 서로 겹치는 모습을 나타내었는데 LRU기법과 2QLRU기법에 따른 성능 차이의 폭이 작기 때문이다. 그림 5(a)와 그림 5(b)에서는 메모리 사이즈가 40% 이상인 구간에서 제안하는 기법의 메모리 적중률이 포화되는 현상을 보여주었다. 이는 여러 애플리케이션들을 짧은 시간 동안 번갈아 사용하는 방법으로 수집된 두 번째 트레이스에는 읽기 및 쓰기 요청이 소수의 특정 페이지에 몰려있어 일정량 이상의 DRAM 공간만 있어도 요청되는 페이지들을 전부 처리할 수 있기 때문으로 분석된다. 또한 그림 5에서 LRU기법과 2QLRU기법의 메모리 적중률은 메모리 사이즈가 80% 이상이 되어서야 포화되는 모습을 나타낸다. 이는 두 기법이 재 참조가 빈번히 발생하는 페이지들을 보조 저장장치로 추출하는 경우가 많기 때문이다. 반면 웹 서핑에 의해 발생한 페이지 요청이 주를 이루는 첫 번째 트레이스의 실험결과를 나타낸 그림 4에서는 메모리 사이즈가 10%에서 70%까지 증가함에 따라 제안하는 기법의 메모리 적중률이 서서히 증가하는 모습을 보이며 70% 이상의 메모리 사이즈에서는 포화되는 현상을 보여준다.

또한 포그라운드 애플리케이션의 메모리 적중률을 살펴보면 전체 애플리케이션의 메모리 적중률보다 최대 3% 포인트 가량 성능이 개선되었음을 보여준다(그림 4(c),

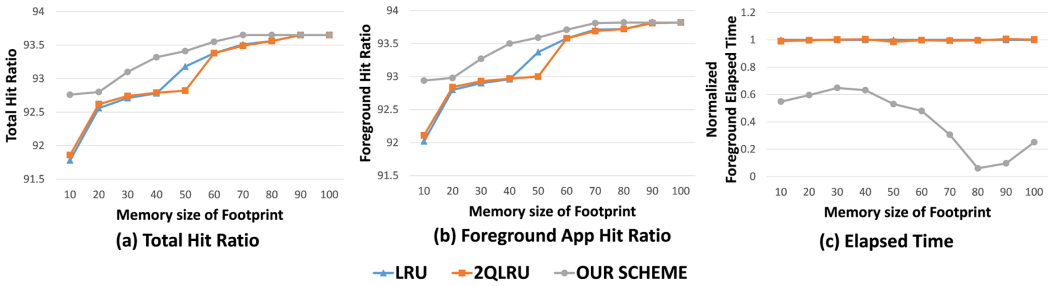

그림 4 트레이스1의 전체 메모리 적중률, 포그라운드 애플리케이션의 수행시간 및 메모리 적중률
Fig. 4 Experiment result with trace1 (foreground application chrome with Hangout and Gmail background application)

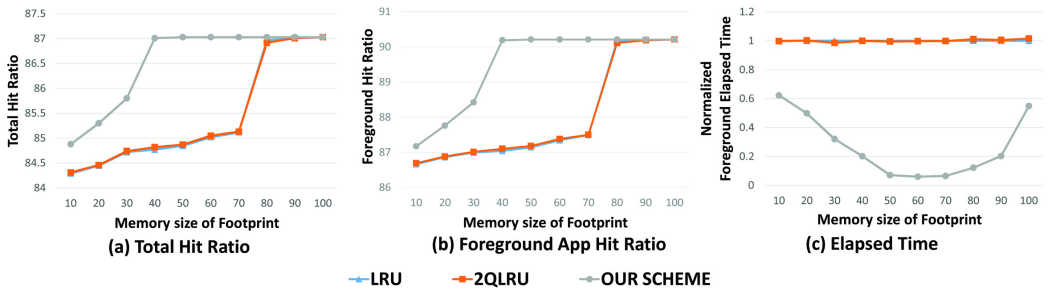

그림 5 트레이스2의 전체 메모리 적중률, 포그라운드 애플리케이션의 수행시간 및 메모리 적중률
Fig. 5 Experiment result with trace1 (mixed trace with various foreground application)

그림 5(c)). 이는 포그라운드 애플리케이션 우대 정책으로 인하여 포그라운드 애플리케이션의 페이지들이 버퍼에 오랫동안 남아있기 때문이다. 또한 사용자가 느끼는 성능에 직접적으로 영향을 끼치는 포그라운드 애플리케이션의 메모리 적중률이 상승함으로써 사용자가 느끼는 체감 성능이 향상될 수 있다. 포그라운드 애플리케이션의 수행시간은 각각의 트레이스에서 LRU 기법의 수행시간과 비교하여 평균 58%, 73% 감소하였고 특히 메모리 크기가 커지면서 급격히 감소하는 형태를 나타내었다. 이는 LRU와 2QLRU는 애플리케이션의 구분 없이 LRU 정책에 따라 페이지들을 추출하는 반면 제안하는 기법에서는 포그라운드 페이지를 우대함에 따라 스토리지로 추출되는 포그라운드 페이지가 급격하게 줄었기 때문이다. 이때 전체 시간을 계산하는 방법은 표1을 바탕으로 DRAM의 읽기 및 쓰기 시간을 1, PCM의 읽기 시간을 2, PCM의 쓰기 시간을 50, 저장장치로 추출되는 시간을 1000의 비율로 가정하고 계산하였다. PCM의 읽기/쓰기 시간을 2와 50으로 설정한 이유는 최대한 PCM의 성능을 낮게 설정하여도 비교 기법들과의 성능 경쟁에서 우위에 있음을 보이고자 함이다. 저장장치로 추출되는 페이지의 숫자는 포그라운드 애플리케이션에 의해서 추출되는 페이지와 백그라운드 애플리케이션에 의하여 추출되는 포그라운드 페이지들의 총 합으로 계산하여 포그라운드 애플리케이션의 독립적인 성능을 측정하였다. 이때 첫 번째 트레이스에 대한 수행 시간 그래프(그림 4(c))의 10%~50% 구간에서 제안하는 기법의 수행시간이 상승하다가 감소하는 구간이 나타난다. 이는 해당 구간에서 제안기법의 수행시간 감소 비율(기울기)이 다른 두 가지 기법의 감소 비율보다 작고 이것을 정규화하여 표현하였기 때문이다. 실제로 메모리 크기에 따른 수행시간 감소 그래프는 선형적으로 나타나며 제안하는 기법은 비교 기법들보다 x축의 모든 구간에 대해 짧은 수행시간을 나타내었다. 메모리 사이즈가 100%인 구간에서는 제안하는 기법의 수행시간이 크게 상승하는 그래프를 볼 수 있다. 이는 메모리 사이즈가 충분히 클 경우 보조저장장치로 추출되는 페이지가 발생하지 않아 비교 기법들과의 수행 시간 차이가 작아지기 때문이다.

그림 6과 그림 7은 두 가지 워크로드를 수행함에 있어서 메모리 사이즈에 따른 에너지 소모량을 표 1에서 나타난 값을 기준으로 계산한 값을 나타낸 그래프이다. 각 그림에서 x축은 메모리 사이즈를 나타내며 y축은 LRU 기법의 소모 에너지를 기준으로 정규화한 2QLRU와 제안한 기법의 에너지 소모량을 나타낸다. 제안한 기법에 의한 에너지 소비량은 LRU 기법의 에너지 소비량을 기준으로 두 가지 워크로드에서 각각 22%, 23% 포

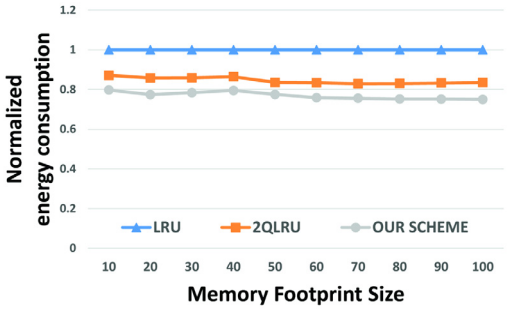

그림 6 트레이스1에 따른 소모 에너지
Fig. 6 Consumed energy by trace1

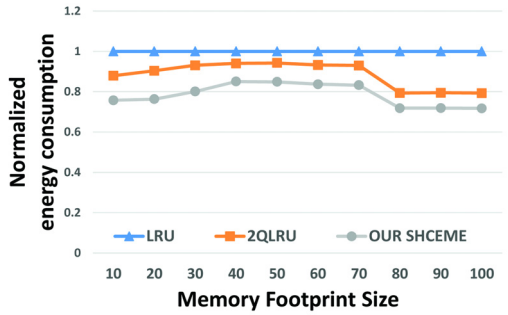

그림 7 트레이스2에 따른 소모 에너지
Fig. 7 Consumed energy by trace2

인트 가량 낮은 결과를 나타내었다. 이는 빈번한 페이지 요청이 발생하는 포그라운드 페이지가 DRAM에 분포되어 있어 PCM의 쓰기 횟수가 다른 기법들보다 적기 때문이다. 이때 세가지 기법 모두 동일한 하이브리드 메모리 환경을 기준으로 계산하였기 때문에 데이터 유지를 위해 DRAM에서 소모되는 전력은 고려하지 않는다.

제안하는 기법에서 백그라운드 애플리케이션의 페이지들은 DRAM을 거치지 않고 곧바로 PCM에 쓰여지게 되므로 비교한 두 가지 기법에 비해 PCM의 쓰기 사용량이 많아 보일 수 있다. 하지만 백그라운드 애플리케이션들은 페이지 요청을 적게 발생시키며 빈번한 페이지 요청이 발생되는 포그라운드 애플리케이션의 페이지들은 PCM으로 추출되지 않고 DRAM에 남아 있게 된다. 이로써 DRAM에서 PCM으로 이동하는 페이지의 총 양이 줄어들게 되며 세가지 기법 중 가장 낮은 에너지를 사용하여 동일한 워크로드를 수행할 수 있었다.

5. 결론 및 향후 연구

본 논문에서는 모바일 디바이스의 전력소모 개선을 위해 DRAM과 PCM을 이용한 하이브리드 메모리 시스템 기반의 버퍼캐시 정책을 제안하였다. 제안한 기법에서는 포그라운드 애플리케이션 페이지들이 높은 우선순위를 갖고 DRAM에서 동작하게 함으로써 동작속도가 느린 PCM의 영향을 최소화하여 사용자가 느낄 수 있는 성능 저하가 발생하지 않도록 하였다. 또한 전체 메모리 적중률과 포그라운드 애플리케이션의 메모리 적중률 성능은 LRU나 2QLRU 대비 성능저하 없이 소폭 향상되었을 뿐만 아니라 PCM의 비율을 전체 메모리의 95% 사용함에도 애플리케이션 수행 시간이 두 가지 워크로드에서 기존 기법과 비교하여 평균 58%, 73% 감소되는 것을 확인 할 수 있었다. 또한 포그라운드 애플리케이션 우대 정책을 사용함으로 인해 PCM에서 발생하는 쓰기 횟수가 줄어들어 전체 메모리 시스템의 소비 전력도 평균 23% 감소하였다.

앞선 실험을 통하여 소량의 DRAM만으로도 기존 시스템과 비교하여 동등수준 이상의 성능을 갖는 버퍼 캐시 기법이 구현될 수 있음을 실험을 통해 확인하였다. 향후 안드로이드 시스템에서 사용하는 SQLite에 의해 발생하는 sync 오퍼레이션의 큰 오버헤드를 줄이는 방법으로써 PCM의 비휘발성 특성을 이용한 메모리 관리 기법에 대해 연구를 진행할 계획이다.

References

[1] A. Carroll and G. Heiser, "An Analysis of Power Consumption in a Smartphone," *Proc. of the USENIX Annual Technical Conference*, pp. 1-14, Jun. 2010.

[2] S. K. Datta, C. Bonnet, and N. Nikaein, "Android Power Management: Current and Future Trends," *Proc. of the Enabling Technologies for Smartphone and Internet of Things*, pp. 48-53, Jun. 2012.

[3] G. Lim, C. Min, D. H. Kang, and Y. I. Eom, "User-Aware Power Management for Mobile Devices," *Proc. of Global Conference on Consumer Electronics*, pp. 151-152, Jan. 2013.

[4] S. J. Han, D. H. Kang, and Y. I. Eom, "Low Power Killer: Extending the Battery Lifespan by Reducing I/O on Mobile Devices," *Proc. of the IEEE International Conference on Consumer Electronics*, 2015, pp. 579-580, Jan. 2015.

[5] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable High Performance Main Memory System Using Phase-Change Memory Technology," *Proc. of the International Symposium on Computer Architecture*, pp. 24-33, Jun. 2009.

[6] S. Lee, H. Bahn, and S. H. 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, Apr. 2013.

[7] G. Dhiman, R. Ayoub, and T. Rosing, "PDRAM: A Hybrid PRAM and DRAM Main Memory System,"

Proc. of the Design Automation Conference, pp. 664-669, Jul. 2009.
[8] S. Eilert, M. Leinwander, and G. Crisenza, "Phase Change Memory: A New Memory Technology to Enable New Memory Usage Models," *Proc. of the International Memory Workshop*, pp. 1-2, May 2009.
[9] N. Gandhewar and R. Sheikh, "Google Android: An Emerging Software Platform for Mobile Devices," *Journal of Computer Science and Engineering*, pp. 12-17, Jan. 2011.
[10] Android Open Source Project [Online]. Available: https://source.android.com/devices/tech/power/index.html (downloaded 2015, May 12)
[11] Android Open Source Project [Online]. Available: https://source.android.com/devices/#Binder IPC (downloaded 2015, May 12)
[12] Android Open Source Project [Online]. Available: https://source.android.com/devices/tech/ram/low-ram.html (downloaded 2015, May 12)
[13] S.-L. Chu, S.-R. Chen, and S.-F. Weng, "Design a Low-Power Scheduling Mechanism for a Multicore Android System," *Proc. of the Parallel Architectures, Algorithms and Programming*, pp. 25-30, Dec. 2012.
[14] Android Open Source Project [Online]. Available: https://source.android.com/devices/tech/power/index.html (downloaded 2015, May 12)
[15] Android Open Source Project [Online]. Available: https://developer.android.com/guide/components/processes-and-threads.html (downloaded 2015, May 12)
[16] T. Johnson and D. Shasha, "2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm," *Proc. of the 20th International Conference on Very Large Data Bases*, pp. 439-450, Sep. 1994.

오 찬 수
2008년 연세대학교 의용전자공학과 학사 2008년~현재 한화테크윈 선임연구원 재직. 2008년~현재 성균관대학교 정보통신대학 석사과정. 관심분야는 스토리지 시스템, 운영체제

강 동 현
정보과학회논문지
제 42 권 제 1 호 참조

이 민 호
2013년 성균관대학교 전자전기공학부 학사. 2013년~현재 성균관대학교 전자전기컴퓨터공학과 석사과정. 관심분야는 운영체제, 파일시스템, 가상화 기술

엄 영 익
정보과학회논문지
제 42 권 제 1 호 참조