논문 2012-50-3-10

고성능 대용량 플래시 메모리 저장장치의 효과적인 매핑정보 캐싱을 위한 적응적 매핑정보 관리기법

(Adaptive Mapping Information Management Scheme for High Performance Large Sale Flash Memory Storages)

이 용 주*, 김 현 우**, 김 희 정**, 허 태 영**, 정 상 혁**, 송 용 호***

(Yongju Lee, Hyunwoo Kim, Huijeong Kim, Taeyeong Huh, Sanghyuk Jung, and Yong Ho Song)

요 약

모바일 디바이스, PC, 서버 형 워크스테이션 시스템에서 널리 사용되고 있는 낸드 플래시 메모리는 기존의 하드 디스크에 비해 저 전력 소비, 높은 성능, 랜덤 접근 가능 등의 장점을 갖는 반면, 덮어쓰기가 불가능하여 데이터를 쓰기 전에는 항상 삭제 연산을 필요로 하는 구조적 약점을 지니고 있다. 이를 극복하기 위해 낸드 플래시 메모리의 제어기는 FTL을 사용하여 디바이스 내부 연산을 변형시킨다. 하지만 고성능 대용량 낸드 플래시 메모리 저장장치의 사용이 증가됨에 따라, 제한된 DRAM 크기에 비해 매핑 알고리즘에서 사용되는 매핑 테이블의 크기가 증가하는 문제가 발생한다. 본 논문은 이러한 DRAM의 용량 부족 문제를 해결하기 위해, 페이지 매핑 기법을 바탕으로한 적응적 매핑정보 캐싱 기법을 제안한다. 적응적 매핑정보 캐싱 알고리즘은 다양한 워크로드 분석을 기반으로 낸드 플래시 접근을 최소한으로 하는 매핑정보 캐싱 방식을 사용한다. 트레이스 기반 시뮬레이터를 통해 실험한 결과, 본 논문에서 제시하는 적응적 매핑정보 캐싱 알고리즘은 기존의 고정 매핑정보 캐싱 알고리즘에 비해 최소 7%에서 최대 70%의 성능향상을 보임을 확인할 수 있었다.

Abstract

NAND flash memory has been widely used as a storage medium in mobile devices, PCs, and workstations due to its advantages such as low power consumption, high performance, and random accessability compared to a hard disk drive. However, NAND flash cannot support in-place update so that it is mandatory to erase the entire block before overwriting the corresponding page. In order to overcome this drawback, flash storages need a software support, named Flash Translation Layer. However, as the high performance mass NAND flash memory is getting widely used, the size of mapping tables is increasing more than the limited DRAM size. In this paper, we propose an adaptive mapping information caching algorithm based on page mapping to solve this DRAM space shortage problem. Our algorithm uses a mapping information caching scheme which minimize the flash memory access frequency based on the analysis of several workloads. The experimental results show that the proposed algorithm can increase the performance by up to 70% comparing with the previous mapping information caching algorithm.

Keywords : 낸드 플래시 메모리, FTL, 매핑 정보, 캐싱 알고리즘
NAND flash memory, flash translation layer, mapping information, caching algorithm

* 정회원, LIG 넥스원
(LIG Nex1)
** 학생회원, 한양대학교 전자컴퓨터통신공학과
(Dept. of Electronics and Computer Engineering, Hanyang University)
*** 정회원, 한양대학교 융합전자공학부
(Dept. of Electronic Engineering, Hanyang University)
※ 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 육성지원 사업의 연구결과로 수행되었음
(NIPA-2012-H0301-12-1011)
접수일자:2012년12월26일, 수정완료일:2013년2월27일

Ⅰ. 서 론

메모리 반도체 기술의 발전에 기인하여, 하드 디스크(HDD: Hard Disk Drive)를 대체할 수 있는 저장장치로 낸드 플래시 메모리가 각광받고 있다. 최근에는 스마트폰, 태블릿과 같은 모바일 디바이스뿐만 아니라, 서버와 노트북과 같은 고성능, 대용량 시스템의 저장장치로 확장되어 그 사용이 증가하고 있다. 낸드 플래시 메모리가 갖는 다음과 같은 장점들이 이러한 발전을 가능하게 하였다. 첫째, 낸드 플래시 메모리는 빠른 임의 읽기 속도를 제공한다. 플래터의 회전과 헤드암의 이동을 통해 원하는 주소에 접근하는 HDD와 달리, 낸드 플래시 메모리는 기계적인 구조를 가지고 있지 않기 때문에 HDD에 비해 임의 데이터 접근 시간이 짧다. 둘째, 낸드 플래시 메모리는 기계적인 구조를 갖지 않기 때문에 외부 충격에 대해 강하다. 셋 째, 낸드 플래시 메모리는 대기 전압을 소모하지 않으므로 낮은 소비전력을 갖는다.

하지만 낸드 플래시 메모리는 최소 저장 단위인 셀의 특성으로 인해 덮어쓰기(In-place update)가 불가능하며, 따라서 데이터를 기록하기 전에는 반드시 해당 페이지가 속한 블록을 삭제해야 한다. 또한 블록마다 셀에 데이터를 쓰고 지우는 횟수(Program/Erase cycle)가 제한되어 있기 때문에, 낸드 플래시 메모리의 내구성 향상을 위한 메커니즘이 필요하다. 이를 극복하기 위해 *Flash Translation Layer(FTL)*라는 소프트웨어 계층을 사용하며, FTL은 논리주소와 물리주소를 일대일로 매핑시킨다. 논리주소는 호스트로부터 저장장치 접근에 사용되는 주소를 의미하고, 물리주소는 낸드 플래시 메모리의 실제 접근 주소를 나타낸다. 호스트로부터 덮어쓰기 명령이 요청되면, 임의의 새로운 물리 페이지를 할당하여 요청된 데이터를 저장 한 후 해당 매핑 정보를 수정한다.

논리주소와 물리주소를 매핑하는 기법은 매핑되는 주소 영역의 단위에 의해 결정되며, 다음과 같이 세 가지 방식으로 구분된다: *페이지 매핑(Page mapping)*, *블록 매핑(Block mapping)*, 그리고 *하이브리드 매핑(Hybrid mapping)*이다. 페이지 매핑은 페이지 단위로 매핑 테이블을 구성하는 방법으로써, 매핑 테이블의 크기가 상대적으로 큰 반면에 빠른 임의 접근을 가능하게 한다. 블록 매핑은 블록 단위 매핑 테이블을 구성하여 매핑 테이블 크기가 작지만, 블록 내에 정해진 오프셋(Offset)을 통해 페이지 주소에 접근하기 때문에 임의 접근에 대한 유연성이 떨어진다. 하이브리드 매핑은 로그 블록과 데이터 블록을 구분하여 페이지 매핑을 적용하는 로그 블록 영역과 블록 매핑을 적용하는 데이터 블록 영역을 하께 사용하는 기법이다.

낸드 플래시 메모리를 사용하는 분야가 다양해지고, 고성능 낸드 플래시 메모리 저장장치에 대한 요구가 증가함에 따라 임의 접근 성능이 뛰어난 페이지 매핑 기법의 필요성이 증대되었다. 하지만 페이지 매핑 기법은 매핑 테이블의 크기가 비대해 진다는 단점과 함께, 저장장치의 대용량화로 인한 매핑 테이블의 크기 변화에 더욱 민감하게 반응하는 문제가 존재한다. 이에 따라 페이지 매핑 기법 적용 시 DRAM 버퍼에 가중되는 부하를 줄이기 위해 매핑 테이블을 *캐싱(Caching)*하는 방법이 제안되었고, 관련 연구가 활발히 진행 중이다. 기존의 캐싱 알고리즘들은 매핑 테이블이 포함된 맵 페이지를 캐싱할 때 그 크기를 고정적으로 사용하였지만, 이는 트레이스에 따라 성능 격차가 큰 단점을 지닌다. 이를 극복하기 위한 방법으로, 본 연구에서는 고성능 낸드 플래시 메모리 기반 저장장치에 적합한 적응적 맵 페이지 캐싱 기법을 제안한다.

본 논문의 구성은 다음과 같다. 제 Ⅱ장에서 기존의 매핑 정보 캐싱 알고리즘을 살펴본다. 제 Ⅲ장에서는 다양한 어플리케이션의 저장장치 접근 패턴을 분석하고, 분석 결과를 바탕으로 적합한 캐싱 알고리즘을 고안한다. 또한 저장장치 접근 패턴에 따라 적응적으로 알고리즘을 선택할 수 있는 적응적 캐싱 알고리즘을 제안한다. 제 Ⅳ장에서는 실험을 통하여 제안된 알고리즘의 성능을 측정하고, 마지막으로 Ⅴ장에서는 결론을 맺는다.

Ⅱ. 관련 연구

지난 수년간, 페이지 매핑 방식을 활용하기 위한 방법으로 맵 페이지 캐싱 기법에 대한 다양한 연구가 진행되어 왔다. 페이지 매핑 테이블의 지역성을 고려하여 DRAM 버퍼의 캐싱 효율을 높이거나, 매핑 테이블의 크기 자체를 줄이기 위한 알고리즘들이 제안되었다. 대표적인 두 가지 매핑 테이블 캐싱 기법은 다음과 같다.

DFTL(Demand-based caching of page mapping table) 기법[1]은 DRAM 버퍼의 부하를 줄이기 위해 *시간적 지역성(Temporal locality)*을 이용하여 페이지 매핑 테이블의 일부만을 버퍼에 캐싱한다. 이와 같은 방법으로 캐싱되는 매핑 테이블의 크기를 줄이고 버퍼의

효율성을 향상시킬 수 있지만, 시간적 지역성만을 고려한 매핑 테이블 캐싱은 캐시 적중률이 낮아지는 단점이 존재한다.

CFTL(Convertible FTL) 기법[2]은 페이지 매핑 기법과 블록 매핑 기법을 동시에 사용하는 하이브리드 매핑 기법에서 각 매핑 테이블을 구분하여 캐싱한다. 호스트로부터 요청된 데이터에 대해서는 페이지 매핑을 적용하고, 요청 횟수에 따라 *핫 데이터(Hot Data)*와 *콜드 데이터(Cold Data)*로 구분한다. 무효한 상태인 페이지들을 수집하여 물리적으로 제거하는 *가비지 컬렉션(Garbage Collection)*이 발생하면 구분된 데이터 중 콜드 데이터를 페이지 매핑에서 블록 매핑으로 매핑 기법을 전환한다. 이러한 매핑 변환 방식은 시간적으로 접근성이 낮은 데이터들의 매핑 유지비용을 최소화할 수 있지만, DFTL과 마찬가지로 시간적 지역성을 고려하여 매핑정보를 캐싱하므로 캐시 적중률이 낮아지는 단점을 지닌다.

Ⅲ. 본 문

기존의 페이지 매핑 테이블 캐싱 알고리즘들은 고정된 크기의 캐싱 윈도우를 사용한다. 이는 시간적 지역성과 *공간적 지역성(Spatial Locality)* 모두 고려하거나 호스트로부터 랜덤 쓰기 요청을 많이 받는 경우, 캐시 미스율이 증가한다. 또한, 불필요한 매핑 정보가 캐싱되어 DRAM영역을 점유하게 될 가능성이 높은 단점이 있다. 이와 같이 호스트의 저장장치 접근 요청에 대한 분석 없이 고정된 캐싱 윈도우를 사용할 경우, 캐시 적중률 향상에 한계가 있다. 본 연구에서는 다양한 워크로드 분석을 통해 적응적 캐싱 윈도우 사용이 가능한 알고리즘을 제안한다.

1. 매핑 테이블 캐싱 알고리즘 성능 분석 방법

매핑 정보 캐싱은 낸드 플래시 메모리 기반 저장장치에 사용되는 DRAM 버퍼의 부하를 줄이기 위한 방법으로서, 효과적인 캐싱 기법에 대한 다양한 연구가 진행 중이다. 하지만 매핑 테이블 캐싱 알고리즘의 성능 측정에 있어 그 기준이 명확하지 않아 시스템 환경에 적합한 알고리즘 선택의 어려움이 있다. 캐시 알고리즘의 성능을 정확하게 예측하기 위해서는 캐시 적중률과 캐시 미스 손해시간을 측정하여 분석하는 과정이 필요하다.

가. 캐시 적중률과 프리캐싱

캐시 적중률은 저장장치 접근 요청에 의해 생성된 매핑 정보와 캐싱된 매핑 정보가 버퍼 상에 존재하는 비율이다. FTL request의 크기에 따라 생성되는 매핑 정보가 다르기 때문에, FTL request 단위로 측정되며 매핑 테이블에 접근한 횟수를 기준으로 나타낸다.

기본적인 매핑 테이블의 캐싱 동작의 정의와 *프리캐싱(Pre-Caching)*의 효과는 다음과 같다. FTL에서 호스트의 request가 FTL request로 변환됨과 동시에 버퍼의 캐싱된 매핑 테이블에 요청한 논리 페이지 주소가 존재하는지 확인한다. 캐싱된 매핑 테이블에 논리 페이지 주소가 존재하지 않는다면 해당 매핑 정보를 낸드 플래시 영역에서 캐싱한다. 만약 주변 주소 값에 대한 매핑 정보를 프리캐싱한다면, 첫 번째 FTL request는 캐시를 탐색하여 적중인지 미스인지 확인하여야 하지만 그 후 하나의 호스트 request에 연관된 FTL request에 대한 매핑 정보는 이미 캐싱된 상태일 가능성이 높다.

나. 캐시 미스 손해시간

캐시 미스 손해시간은 캐시 미스가 발생하였을 때, 해당 매핑 정보를 낸드 플래시 메모리에서 읽어옴으로써 지연되는 시간을 의미한다. 캐시 미스 소비시간은 식1 과 같이 캐시 미스로 인한 낸드 플래시 읽기시간(T_{rm}: Read time for miss), 쫓겨나기 동작을 위한 낸드 플래시 쓰기시간(T_{pf}: Program time for flush), 쫓겨나기 동작에서 생기는 부가적인 낸드 플래시 읽기시간(T_{rf}: Read time for flush)의 합으로 나타낸다. 각각의 시간은 식 2, 3, 4를 통해 계산될 수 있다.

$$T_{penalty} = T_{rm} + T_{pf} + T_{rf} \quad (1)$$

$$T_{rm} = t_R + t_{RC} \times (\text{읽을 데이터/플래시 메모리 } I/O \text{폭}) \quad (2)$$

$$T_{pf} = t_{PROG} + t_{WC} \times (\text{쓸 데이터/플래시 메모리 } I/O \text{폭}) \quad (3)$$

$$T_{rf} = t_R + t_{RC} \times (\text{읽을 데이터/플래시 메모리 } I/O \text{폭}) \quad (4)$$

2. Spatial Locality Effect (SLE)

캐싱하는 매핑정보의 크기와 접근 위치를 결정하기 위해 맵 페이지 접근에 대한 공간적 지역성이 존재하는 범위*(SLE: Spatial Locality Effect)*를 분석한다. SLE 분석 결과를 바탕으로 적합한

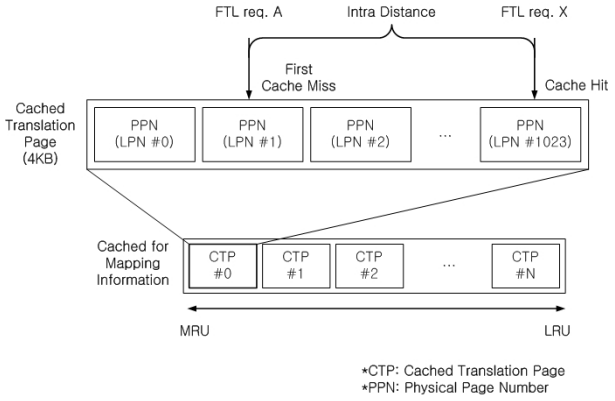


그림 1. Intra-distance 측정 방법
Fig. 1. Illustration of measuring Intra-distance.

캐싱 크기 및 알고리즘을 발견할 수 있다. 동일 맵 페이지에 접근하는 근접 리퀘스트 간의 거리(Intra-distance)를 측정하여 이를 기준으로 근접 리퀘스트 간의 연관성을 분석한다. 그림 1의 예에서, FTL request A는 LPN #1을 시작으로 저장장치에 접근하는 명령이다. 따라서, 최초 *주소변환 페이지(TPN: Translaation Page Number)*를 캐싱 하는 동기가 된 LPN #1과 새로운 FTL request X에 대한 논리 주소 LPN #1023 간의 거리가 해당 맵 페이지 접근에 대한 Intra-distance 이다.

Intra-distance와 캐싱 효과의 연관성 증명을 위한 실험을 위해, 4MB DRAM 버퍼와 페이지 단위 매핑 테이블 캐싱 알고리즘을 가정하여 SLE를 측정한다. 그림 1의 예에서 FTL request A에 대한 매핑 정보가 캐싱되어 있지 않아 캐시 미스가 발생한다. 따라서 FTL request A로 인해 LPN #1에 할당된 물리 페이지 주소를 포함하는 TPN이 캐싱된다. 캐싱 시에는 동적 방지 매커니즘이 동작하게 되는데, 동적 방지 매커니즘이란 프리 캐싱 되는 매핑 정보들로 인해 실제 접근이 발생한 매핑 정보가 맵 캐시에서 쫓겨나는 경우를 방지하기 위해 실제 접근한 주소에 대한 매핑 정보를 맵 캐시의 *MRU(Most Recently Used)*로 옮기는 일련의 과정이

표 1. 낸드 플래시 메모리 특성
Table 1. Features of NAND Flash memory used in the experiment.

특성		값
페이지 크기		4Kbytes
읽기 시간	tRC	25ns
	tR	200㎲
쓰기 시간	tWC	25ns
	tPROG	1.6㎳

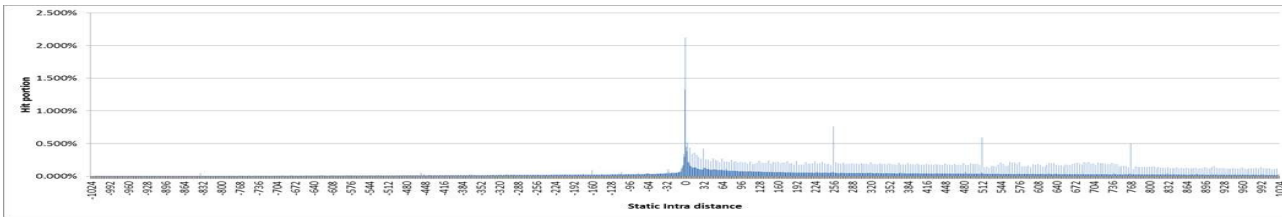
다. 그 후 새로운 FTL request X가 요청되면 요청한 물리 주소에 대한 정보를 할당한 LPN #1023가 CTP #0에 있기 때문에 캐시 히트가 발생한다. 워크로드를 사용하여 모든 동작요청을 실행한 후 추출한 Intra-distance의 합을 구하면 그 결과값은 SLE를 나타낸다.

SLE 측정 시뮬레이션에 사용된 낸드 플래시 메모리는 표 1과 같은 특성을 가지며, 접근 패턴을 분석할 워크로드의 특성은 표 2와 같다. PC 트레이스는 PC 작업 환경에서 인터넷 서핑, 문서 작성, 윈도우 탐색기를 실행하여 추출한다. TPCC 트레이스[4]는 On-Line Transaction Processing(OLTP) 환경에서 서버 시스템의 성능평가를 위해 사용되는 벤치마크로, Hammerora 툴을 사용해서 가상의 ware-house와 가상 유저를 설정하여 추출한다. Financial 트레이스[5] 또한 OLTP 환경에서 서버 시스템 성능 평가를 위한 벤치마크이다. 표 2 트레이스 특성에서 Address Range는 낸드 플래시 메모리가 접근할 수 있는 주소 값의 영역을, Actual Address Range는 실제 접근한 가장 작은 주소 값과 가장 큰 주소 값의 차를, Actual Request Range는 실제로 접근한 주소영역을 의미한다.

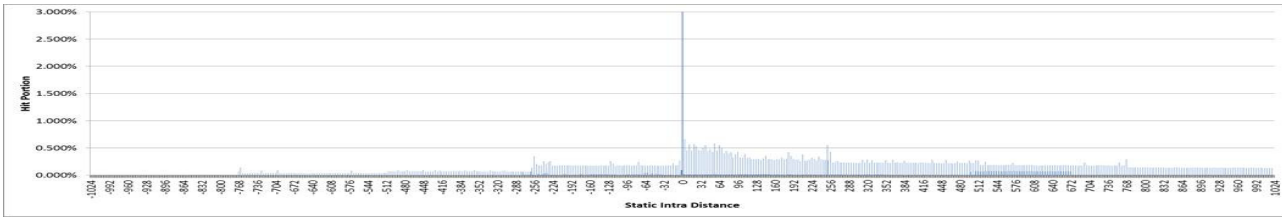
그림 2는 다양한 트레이스를 통해 추출한 SLE 결과 중 트레이스의 특성을 대표할 수 있는 SLE 측정 결과이다. 가로축은 Intra-distance값으로 페이지 단위(4KB)의 맵 페이지 캐싱을 가정하였기 때문에 그 범위는 -1023부터 1023(4KB/4B-1)까지가 된다. 세로축은 전체

표 2. 워크로드 특성
Table 2. Characteristics of selected real-world workloads.

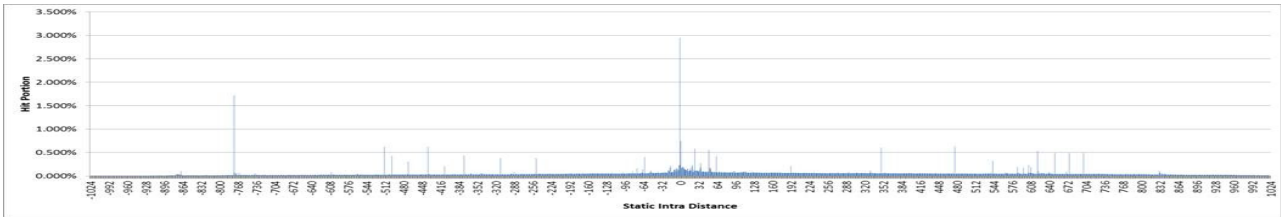
Traces	Address Range [GB]	Actual Address Range[GB]	Actual Request Range[GB]	# of OS Req.	# of FTL Req.	R/W Ratio [%]	Ave. of Req. size [KB]
PC	780	778.2	11.7	253,773	1,537,515	67.8/32.2	21
TPC-C	60	32.0	2.6	271,707	3,094,854	7.9/92.1	46
TPC-C	60	32.0	2.6	292,948	2,677,233	12/88	37
Financial	16	3.7	0.5	5,334,946	9,156,818	0/100	3



(a) PC 2



(b) TPC-C 3



(c) Financial 1

그림 2. SLE 측정 결과
Fig. 2. Results in measuring SLE.

Intra-distance 값들에 대한 해당 Intra-distance 빈도의 백분율 값을 나타낸다.

그림 2-(a), (b), (c) 모두 Intra-distance의 값이 0인 경우가 가장 많다. 이는 트레이스의 저장장치 접근이 높은 시간적 지역성을 가짐을 의미한다. 그림 2-(a), (b)는 Intra-distance가 0인 값에서부터 양의 방향으로 접근이 많은 것을 확인 할 수 있다. 이는 최초 동작 요청에 대한 양의 방향으로의 공간적 지역성이 높음을 의미한다. PC 트레이스의 경우, 최초 FTL request로부터의 Intra-distance 값이 작음을 그림 2-(a)를 통해 알 수 있다. 따라서 캐싱 윈도우 사이즈를 작게 하는 것이 유리하다. TPCC 트레이스의 경우, Intra-distance 값이 큰 FTL request가 다수 요청된 것을 그림 2-(b)를 통해서 확인할 수 있다. 그러므로 캐싱 윈도우 사이즈를 크게 하는 것이 유리함을 추측할 수 있다. Financial 트레이스의 경우, 강한 랜덤접근 특성으로 인해 최초 FTL request를 기준으로 양방향 모두 다양한 접근이 발생한다. 즉, Financial 트레이스는 높은 시간적 지역성과 낮은 공간적 지역성을 갖는다.

3. 매핑 정보 캐싱 윈도우 크기와 성능 분석

캐싱 윈도우의 크기 및 위치를 기준으로 다양한 캐싱 알고리즘을 적용하여 그 성능을 측정하였다. 사용된 알고리즘은 Full-page caching, Half-page caching, 그리고 Forward-prefetching이다.

Full-page caching algorithm은 매핑 정보를 페이지 단위로 캐싱한다. Full-page caching을 사용하여 주소변환 페이지를 DRAM 버퍼에 저장할 시에는 캐싱되는 매핑 주소의 논리주소, 매핑 정보의 변경이 있었는지를 확인하는 *더티 비트(Dirty bit)*, 주소변환 페이지 전체를 저장한다. 이 알고리즘은 DRAM 버퍼에 빈 공간이 없어서 희생 페이지 선택하여 쫓아내기(Flush) 동작 실행 시, 부가적 페이지 읽기 동작이 발생하지 않고 해당 주소변환 페이지만 업데이트 하면 되는 장점이 있다. 그러나 캐싱 단위가 커 캐시의 효율이 낮다.

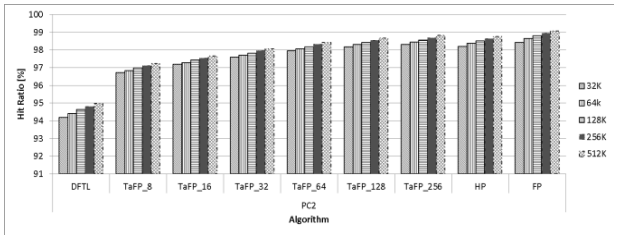
Half-page caching algorithm은 매핑 정보를 반 페이지 단위로 캐싱한다. DRAM 버퍼에 저장되는 형태는 Full-page caching algorithm과 동일하나 주소변환 페이지 전체가 아닌 반 페이지만을 저장한다. 이 알고리즘은 캐싱 단위가 작아 캐시의 효율이 높지만, DRAM

버퍼에 빈 공간이 없어서 희생 페이지 선택하여 쫓아내기(Flush) 동작 실행 시, 다른 반 페이지와 함께 해당 주소 변환 페이지를 업데이트해야 하므로 추가적 읽기 동작이 발생한다.

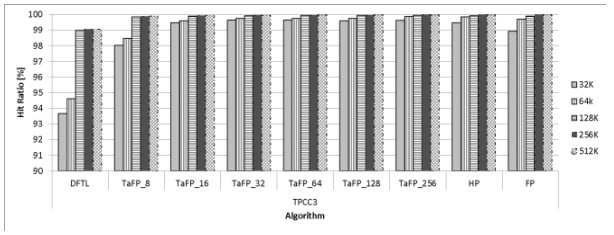
Forward-prefetching algorithm은 3.2절의 Intra-distance 측정 결과를 반영한 방식이다. 그림 2에 나타난 후방 공간적 지역성을 고려하여 프리패칭 윈도우(Prefetching window) 크기만큼 뒤쪽 매핑 정보를 추가 캐싱한다. 프리패칭 윈도우는 8Bytes, 16Bytes, 32Bytes, 64Bytes, 128Bytes, 256Byte 크기로 다양하게 적용이 가능하다. DRAM 버퍼에 저장될 때, 프리패칭 윈도우의 크기 또한 저장한다. 위의 두 알고리즘에 비해 작은 크기의 윈도우를 사용하기 때문에 캐시의 효율성이 높다는 장점을 가지는 반면, Flush 동작 실행 시 추가적 읽기 동작이 발생한다는 단점을 지닌다.

Forward-prefetching algorithm을 사용할 때 프리패칭 윈도우의 크기가 클수록 두 개의 주소변환 페이지를 접근할 확률이 높아진다. 이 경우 불필요한 낸드 플래시 메모리의 읽기, 쓰기가 수행된다. 이를 방지하기 위하여 TPN-aware 매커니즘을 사용하는데, 이 방식은 캐싱하고자 하는 영역 내에 두 개 이상의 변환 페이지가 속한다면 처음 변환 페이지만을 캐싱하는 것이다. 이후 본 논문에서 제시되는 Forward-prefetching algorithm은 TPN-aware 메커니즘을 포함한다.

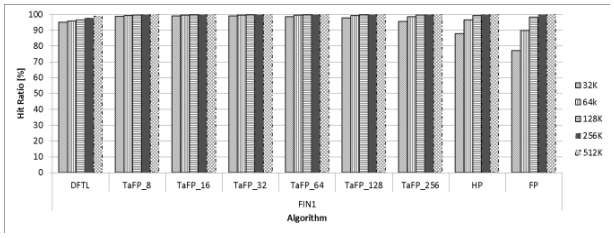
위의 세 알고리즘의 효율을 알아보기 위해 캐시 적중률과 캐시미스로 인한 평균 소비시간을 측정하였다. 표 2에서 제시한 워크로드를 사용하였고, DRAM 버퍼에 캐싱 가능한 매핑 테이블의 크기를 32KB, 64KB, 128KB, 256KB, 512KB로 설정하여 시뮬레이션을 수행하였다. TPN-aware Forward-Prefetching N(TaFP_N), Half-page caching, Full-page caching algorithm과 [1]에서 제시된 DFTL을 비교하였다. TPN-aware Forward-Prefetching algorithm의 N은 프리패칭 윈도우의 크기를 의미한다. 캐싱 가능한 매핑 테이블의 크기를 32KB, 64KB, 128KB, 256KB, 512KB로 설정하여 시뮬레이션을 수행하였다. TPN-aware Forward-



(a) PC trace



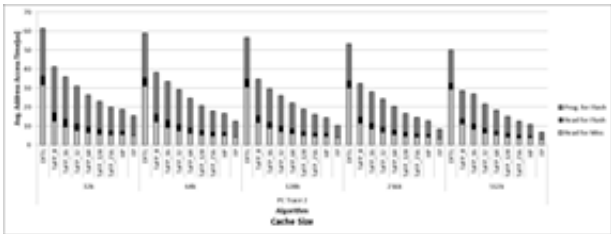
(b) TPC-C trace



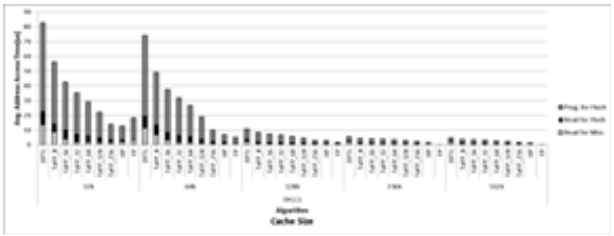
(c) Financial trace

그림 3. 매핑 캐시 크기에 따른 각 워크로드의 알고리즘 별 캐시 적중률

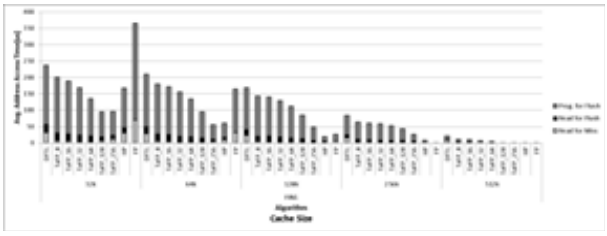
Fig. 3. Cache hit ratio of each algorithm according to the size of mapping cache.



(a) PC trace



(b) TPC-C trace



(c) Financial trace

그림 4. 매핑 캐시 크기에 따른 각 워크로드의 알고리즘 별 평균 캐시 미스 손해시간

Fig. 4. Average cache miss latency of each algorithm according to the size of mapping cache.

Prefetching N(TaFP_N), Half-page caching, Full-page caching algorithm과 [1]에서 제시된 DFTL을 비교하였다. TPN-aware Forward-Prefetching algorithm의 N은 프리패칭 윈도우의 크기를 의미한다.

그림 3은 각 워크로드의 알고리즘 별 캐시 적중률을 나타낸다. 모든 워크로드에서 버퍼의 크기가 증가할수록 캐시 적중률 또한 증가하였다. 하지만 가장 좋은 캐시 적중률을 갖는 알고리즘은 각 워크로드마다 다르다. 캐시 미스로 인해 발생하는 손해($T_{penalty}$: Cache miss penalty)를 측정한 결과는 그림 4와 같다. 그림 4의 각 그래프는 알고리즘 별 FTL request 당 평균 캐시 미스 손해시간을 나타낸다. 이 결과 미스 손해시간이 가장 작은 알고리즘은 각 워크로드마다 다름을 보여준다. 이와 같은 실험 결과로 최적의 맵 캐시 성능을 갖는 알고리즘은 워크로드에 따라 다르며, 이를 반영하는 맵 캐시를 설계하기 위해서는 적응적 캐싱 알고리즘이 필요함을 알 수 있다.

4. 적응적 매핑 정보 캐싱 알고리즘

3.3절의 결과로부터 알 수 있듯이, 다양한 워크로드에 적합한 캐싱 알고리즘을 구현하기 위해서는, 캐싱 윈도우 사이즈를 접근 패턴에 따라 적응적으로 변화하는 기법이 요구된다. 따라서 본 논문에서는 적응적으로 매핑 정보를 캐싱하기 위하여 *DID(Dynamic Intra Distance)*를 측정하고 이를 기준으로 *Score board*를 구성한다. Score board는 각 캐싱 알고리즘에 대한 효과를 Score 형태로 나타낸 표이다. 이 점수를 기준으로 캐싱 윈도우의 크기를 결정한다.

가. DID(Dynamic Intra Distance) 측정

호스트의 접근 패턴에 따라 적응적으로 매핑 정보를 캐싱하기 위해서는 Intra-Distance를 동적으로 점검하는 DID 측정이 필요하다. 제안하는 알고리즘은 DID 값을 기반으로 DRAM 버퍼의 상태를 확인하여 효율이 가장 높을 것으로 예상되는 매핑 캐싱 알고리즘을 선택한다. DID는 FTL request가 DRAM 버퍼에서 매핑 정보를 탐색하는 과정 중에 계산 되고, 캐시 적중 여부에 따라 측정 방법이 달라진다. 캐시 적중인 경우, 접근된 매핑 정보의 마지막 주소와 FTL request와의 거리를 측정한다. 캐시 미스인 경우에는 가장 가까운 매핑 정보의 시작 주소 혹은 마지막 주소를 확인하여 이와 FTL request와의 거리를 측정한다.

나. DID를 기반으로 한 score board mechanism

FTL request의 매핑 정보가 맵 캐시 내에 존재할 때, 해당 매핑 정보가 프리캐싱될 수 있도록 하는 알고리즘에 대해 score를 1씩 증가시킨다. 해당 FTL request의 매핑 정보는 이전에 Full-page caching, TaFP64, TaFP128, TaFP256 알고리즘을 통해 프리캐싱 하였을 경우, 현재 캐시 히트가 발생할 수 있다. 이 때 해당 알고리즘에 대한 score를 1씩 증가시킨다. 이 값을 score board에 반영할 때 오래된 측정값보다 최신의 측정값에 우위를 두기 위해서 오래된 측정값은 1보다 작은 가중치(weight)를 곱하여 저장한다.

다. Balanced score board mechanism

Score board mechanism에서는 FTL request 이후 DRAM 버퍼에 각 매핑 캐싱 알고리즘을 적용했을 경우의 성능을 측정한다. 하지만 각 알고리즘의 캐싱 크기 차이로 인해 DID를 측정했던 매핑 정보가 DRAM 버퍼에서 쫓겨날 수 있다. 따라서 이를 방지하기 위하여 *Balanced score board mechanism*을 사용한다. Balanced score board mechanism은 캐싱된 매핑 정보의 최근 사용된 시점과 현재 DRAM 버퍼에 캐싱된 매핑 정보를 기준으로 각 매핑 캐싱 알고리즘 적용 시 DRAM 버퍼 내 존재 여부를 예측한다. 각 매핑 캐싱 알고리즘을 적용하였을 시에 DID를 측정할 수 있는 범위가 제한된다. DRAM 버퍼에서 가장 적은 수의 주소 변환 페이지 항목을 포함할 것으로 예상되는 Full-page caching 알고리즘의 경우, DID를 측정할 수 있는 범위가 가장 좁다. 반면 가장 많은 수의 주소 변환 페이지 항목을 포함할 것으로 예상되는 TaFP8의 경우, DID 측정 가능 범위가 DRAM 버퍼 전체를 포함한다.

Ⅳ. 실험 및 분석

1. 실험 환경

적응적 매핑 정보 캐싱 알고리즘의 효율을 측정하고 기존의 고정 매핑 정보 캐싱 알고리즘과 비교하기 위해 트레이스 기반 시뮬레이터를 사용하였다. 시뮬레이션에서 호스트의 메모리 접근을 기록한 워크로드를 사용하였으므로, 이를 LPN으로 변환 후 알고리즘을 적용하였다. 각 FTL request에 대해 DID를 측정하고, 이를 기준으로 Balanced Score Board를 업데이트한다.

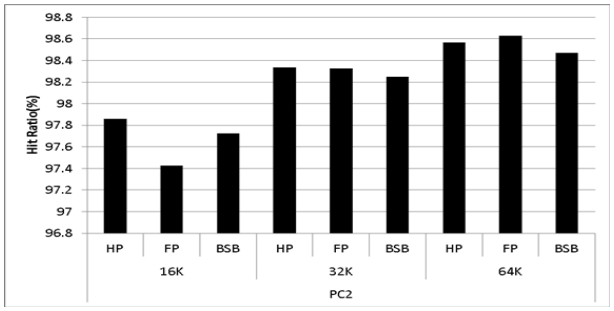
2. 성능 평가

가. 캐시 히트율

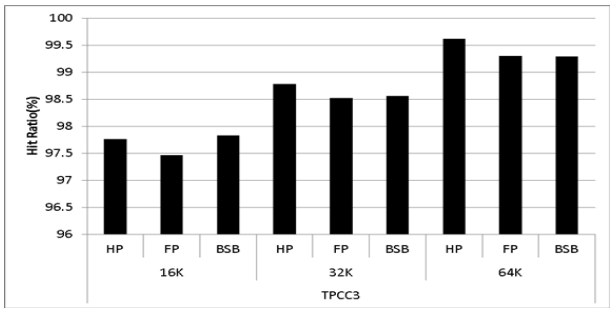
그림 5는 각 워크로드에 대한 Half-page caching, Full-page caching, 그리고 BSB(Balanced Score Board) 알고리즘의 캐시 히트율을 나타낸다. PC 트레이스와 TPCC 트레이스를 사용한 경우, BSB 알고리즘이 Full-page caching에 비해 캐시 히트율이 낮은 결과를 보인다. 이는 PC 트레이스와 TPCC 트레이스의 공간적 지역성이 넓은 범위에 적용되기 때문이다. BSB알고리즘의 히트율은 Financial 트레이스를 사용한 경우 가장 높다. 이는 Financial 트레이스의 OS request의 크기가 커 공간적 지역성이 크지 않고, 시간적 지역성이 높기 때문이다.

나. 캐시 미스 손해 시간

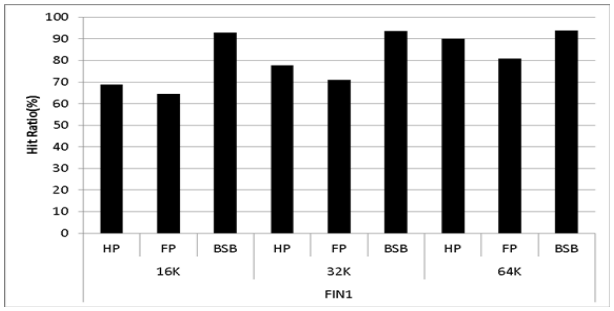
그림 6은 각 워크로드에 대해 FTL request 당 캐시 미스로 소비되는 평균 시간을 나타낸다. 캐시 미스 손해시간을 비교하는 각 알고리즘의 효율은 앞에서 측정한 캐시 히트율을 기준으로 한 알고리즘의 효율과 다른 결과를 보인다. 이는 캐싱 윈도우를 달리하며 캐싱을 진행함으로써 낸드 플래시 메모리에 접근하는 횟수가 각 알고리즘 마다 다르기 때문이다. BSB 알고리즘은 각 구간마다 DRAM 버퍼의 남은 공간을 고려하여 가장 효율적인 캐싱 알고리즘을 선택한다. 각 알고리즘에 대한 BSB 알고리즘의 성능향상 정도는 표 3과 같다.



(a) PC trace

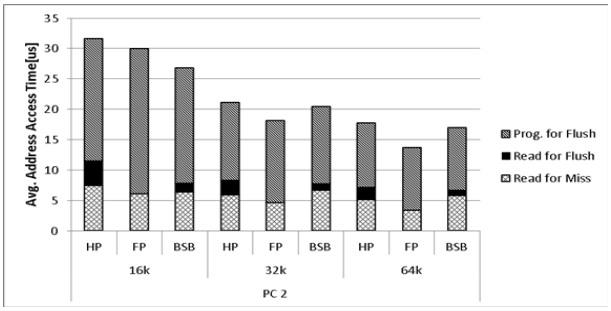


(b) TPC-C trace

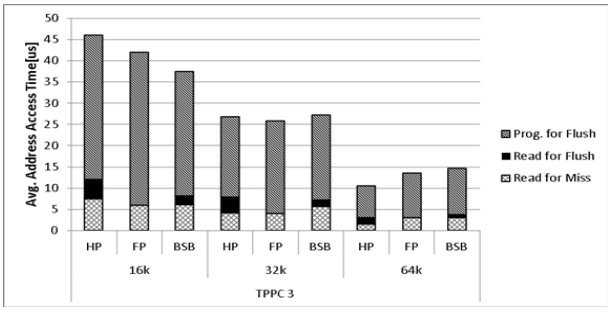


(c) Financial trace

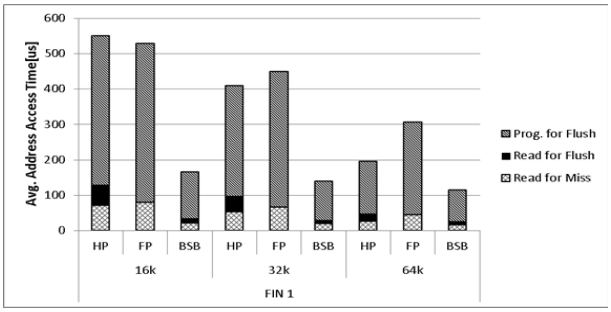
그림 5. 각 워크로드에 대한 캐시 히트율
Fig. 5. Cache hit rate for each workload with different algorithms.



(a) PC trace



(b) TPC-C trace



(c) Financial trace

그림 6. 각 워크로드에 대한 캐시 미스 손해 시간
Fig. 6. Cache miss latency for each workload with each algorithms.

표 3. 각 알고리즘 별 고정 알고리즘 사용 방식과 비교한 BSB 알고리즘의 성능향상 정도
Table 3. Impact of BSB algorithm compared with fixed algorithm in each workload.

Cash Size	16KB		32KB		64KB	
Traces	Vs. Half-page Caching	Vs. Full-page Caching	Vs. Half-page Caching	Vs. Full-page Caching	Vs. Half-page Caching	Vs. Full-page Caching
PC 2	15%	11%	3%	-13%	5%	-24%
TPC-C 3	19%	11%	-2%	-6%	-40%	-8%
Financial 1	70%	69%	66%	69%	41%	63%

성능향상 정도는 캐시 미스 손해 시간의 감소 정도로 표현되었다. 캐시 크기가 16KB인 경우, 모든 워크로드에 대해 BSB가 고정 알고리즘 사용 방식에 비해 최대 70%에서 최소 7%의 성능향상이 있음을 알 수 있다. 32KB 캐시의 경우 일부 PC 워크로드에 대해서만 성능향상이 있었다. 64KB 크기의 캐시에서는 BSB 알고리즘을 사용한 경우 Full-page caching에 비해 성능이 하락하는데, 이는 캐시 크기가 넉넉하기 때문이다. 하지만 캐시의 크기가 충분한 경우에는 매핑 테이블로 인한 DRAM 버퍼의 부하가 크지 않으므로 본 논문에서는 제외한다.

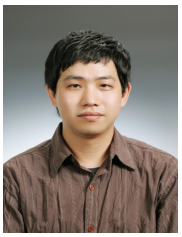
Ⅳ. 결 론

본 논문에서는 호스트의 저장장치 접근 패턴에 따라서 매핑 테이블에 대한 캐싱 크기가 변화하는 적응적 매핑 정보 캐싱 알고리즘을 제시하였다. 제안하는 알고리즘은 호스트의 저장장치 접근 패턴을 다양하게 수용하는 알고리즘을 구현하고, DID 측정과 BSB 매커니즘 적용을 통해 구현하였다. BSB 매커니즘을 사용하였을 경우, 다양한 워크로드에 대해 적응적으로 매핑 캐싱 알고리즘을 사용할 수 있어, 고정 매핑 캐핑 알고리즘에 비해 최소 7%에서 최대 70%의 성능효과가 있음을 확인할 수 있었다. 적응적 매핑 정보 캐싱 알고리즘은 특정 워크로드에서는 최상의 성능을 보이는 고정 매핑 캐싱 알고리즘에 비해 성능이 낮았지만, 평균적으로 거의 모든 워크로드에서 우수한 성능을 보임을 알 수 있었다.

참 고 문 헌

[1] Y. Kim, “DFTL : A Flash Translation Layer Employing Demand-based Selective Caching of Page-level Address Mappings,” pp. 229 - 240, 2009.
[2] S. Jiang, L. Zhang, X. Yuan, H. Hu, and Y. Chen, “S-FTL : An Efficient Address Translation for Flash Memory by Exploiting Spatial Locality.”
[3] D. Park, S. Member, and B. Debnath, “CFTL : An Adaptive Hybrid Flash Translation Layer with Efficient Caching Strategies,” no.September, pp. 1 - 15, 2011.
[4] “Umass Trace Repository: OLTP Application I/O, ” http://traces.cs.umass.edu/index.php/storage/storage.
[5] “The Transaction Processing Performance Council,” http://www.tpc.org.

저 자 소 개



이 용 주(정회원)
2009년 한양대학교 미디어통신공학 졸업(학사).
2011년 한양대학교 전자컴퓨터통신공학과 졸업(석사).
2011년~현재 LIG 넥스원 연구원.
<주관심분야 : 임베디드 시스템, 멀티미디어, 비휘발성 메모리 등>

김 희 정(학생회원)
2012년 한양대학교 전자통신공학/컴퓨터 졸업(학사).
2012년~현재 한양대학교 전자컴퓨터공학과 석사과정.
<주관심분야는 컴퓨터구조, 임베디드 시스템, 낸드 플래시 메모리 등>

정 상 혁(학생회원)
2008년 한양대학교 미디어통신공학/컴퓨터 졸업(학사).
2010년 한양대학교 전자컴퓨터통신공학과 졸업(석사).
2010년~현재 한양대학교 전자컴퓨터공학과 박사과정.
<주관심분야 : 컴퓨터구조, 임베디드 시스템, 비휘발성 메모리 등>

김 현 우(학생회원)
2011년 동의대학교 전자공학 졸업(학사).
2012년~현재 한양대학교 전자컴퓨터공학과 석사과정.
<주관심분야 : 멀티코어 시스템, 멀티미디어 코덱, 임베디드 시스템 등>

허 태 영(학생회원)
2012년 한국해양대학교 전파공학과 졸업 학사),
2012년~현재 한양대학교 전자컴퓨터공학과 석사과정.
<주관심분야 : 컴퓨터 구조, 임베디드 시스템, 낸드 플래시 메모리 등>

송 용 호(정회원)
1989년 서울대학교 컴퓨터공학과 졸업(학사)
1991년 서울대학교 컴퓨터공학과 졸업(석사).
2002년 University of Southern California Electrical Engineering 졸업 (박사).
1991년~1996년 삼성전자 전임연구원.
1996년~2002년 University of Southern California 연구조교.
2002년~2003년 (주)조선 IT 연구소장.
2003년~현재 한양대학교 융합전자공학부 교수.
IEEE International Parallel and Distributed Processing Symposium 프로그램 구성위원.
<주관심분야 : 시스템 구조, SoC, NoC, 멀티미디어, 비휘발성 메모리 등의 임베디드 시스템 구조 등>