낸드 플래시를 위한 온칩 레지스터를 활용한 가비지 컬렉션 부하 감소 기법

(Exploiting On-chip Registers to Reduce the Garbage Collection Overhead for NAND Flash-based Storage Systems)

이 성 진, 김 지 홍*
서울대학교 전기·컴퓨터공학부
(Sungjin Lee, Jihong Kim)
(School of Computer Science & Engineering, Seoul National University)

Abstract : NAND flash memory is widely used as storage devices for many embedded systems. However, due to its erase-before-write feature, NAND flash-based storage systems should support an efficient garbage collection to achieve the high I/O performance. In this paper, we propose a new garbage collection technique that exploits an on-chip register to reduce the garbage collection overhead, improving the I/O performance. By effectively using an on-chip register within a NAND flash chip, our technique reduces data transfer times between a NAND flash chip and a host system dramatically that account for a significant portion of the garbage collection. The experimental results show that the proposed technique can reduce the garbage collection overhead up to 41% under a variety of benchmarks as well as realistic workloads.

Keywords : 낸드 플래시 메모리, 가비지 컬렉션, 플래시 변환 계층, 온칩 레지스터

I. 서 론

NAND 플래시 메모리는 휴대폰, MP3 플레이어, 디지털 카메라 및 캠코더와 같은 임베디드 시스템에서 널리 사용되는 저장 매체이다. 하드디스크와 달리 NAND 플래시 메모리는 내부적으로 블록(block)과 페이지(page)의 형태로 구성되어 있다. 하나의 플래시 칩 내에는 다수의 블록이 존재하고, 각 블록은 다시 여러 개의 페이지들로 나뉜다. 보통 각 페이지의 크기는 2KB이며, 하나의 블록은 64개의 페이지로 구성되어 있다. NAND 플래시 메모리에서 데이터 읽기 및 쓰기는 페이지 단위로 수행된다. 플래시 메모리는 덮어 쓰기를 미지원하기 때문에, 한번 데이터가 기록된 페이지를 다시 사용하기 위해서는 해당 페이지에 대한 삭제 연산을 수행해야한다. 이러한 삭제는 블록 단위로 수행된다. 플래시 칩 내에는 임시 데이터 저장을 위한 페이지 크기의 온칩 레지스터가 포함되어 있으며, 보통 2~4개 정도가 존재한다. 동일한 온칩 레지스터를 공유하는 블록의 집합을 플레인(plane)이라 부른다 [1].

플래시 메모리의 덮어쓰기 제약을 극복하기 위해, 플래시와 파일 시스템 사이에 플래시 변환 계층(Flash Translation Layer, 이하 FTL)이라 불리는 소프트웨어 계층이 사용된다. FTL은 크게 주소 사상과 가비지 컬렉션 두 가지 기능을 지원한다. 주소 사상 기법은 파일 시스템의 논리주소를 플래시의 물리주소로 변환시켜주는 기능을 수행하며, 덮어쓰기 발생 시 해당 데이터를 플래시 메모리 내 빈 페이지에 대신 기록하여 블록 삭제 및 페이지 복사 횟수를 감소시킨다. 이러한 쓰기 전략을 외부갱신(out-place-update)이라고도 한다.

외부갱신 방식은 오래된 데이터를 저장하고 있는 무효화된 페이지를 발생시킨다는 문제점을 지니고 있다. 무효한 페이지가 불필요하게 점유하고 있는 공간을 재이용하기 위해 FTL은 가비지 컬렉션 기능을 지원 한다. 가비지 컬렉션은 블록 내의 최신 데이터를 저장하고 있는 유효한 페이지를 다른 빈 페이지에 복사한 후 해당 블록을 삭제함으로써 무효한 페이지를 가용한 빈 페이지로 변환 시킨다.

※ 본 논문은 BK21사업에 의하여 지원되었으며, 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 (No. R0A-2007-000-20116-0, R33-2008-000-10095-0) 수행되었습니다. 본 연구를 위해 연구장비를 지원하고 공간을 제공한 서울대학교 컴퓨터 연구소에 감사드립니다.

일반적으로 가비지 컬렉션에 따른 페이지 복사는 (1) 해당 페이지의 데이터를 플래시 칩 내의 온칩 레지스터로 읽는 과정과 (2) 온칩 레지스터에 저장된 데이터를 플래시 칩 버스를 이용하여 호스트 시스템의 메모리에 저장하는 과정 (3) 호스트 시스템에서 다시 온칩 레지스터로 데이터를 전달하는 과정 (4) 온칩 레지스터의 데이터를 빈 페이지에 쓰는 과정으로 이루어져 있다. 온칩 레지스터로 페이지 데이터를 읽는 시간은 약 25us 이고, 이를 플래시 페이지에 기록 하는 시간은 200us 정도이다. 반면 플래시 칩 버스를 통한 플래시 칩과 호스트 간 데이터를 전송 시간은 100us이다. 따라서 한 개의 페이지 이주 시 425us의 시간이 걸린다고 할 때, 전체 시간의 약 47% (200us)는 플래시 칩과 호스트 시스템간의 데이터 전달에 소비됨을 알 수 있다.

이러한 데이터 전달 시간을 감추기 위해 플래시 제조사들은 온칩 레지스터만을 활용하여 페이지 이주가 가능하도록 하는 기능을 제공하고 있다 [1]. 본 기능을 활용할 경우 플래시 칩과 호스트간의 데이터 전송 과정이 제거될 수 있다. 다만, 온칩 레지스터를 통한 페이지 복사 기능은 원본 데이터의 블록과 데이터가 쓰일 블록이 동일한 플레인 내에 속한 경우에만 가능하다.

본 논문에서는 온칩 레지스터의 활용을 FTL 단계에서 최대화 시킬 수 있는 효율적인 블록 할당 기법을 제안한다. 제안하는 기법은 가급적이면 가비지 컬렉션에 의한 페이지 이주가 동일한 플레인 내에서 발생하도록 데이터와 블록간의 배치를 변경함으로써 온칩 레지스터의 활용도를 높이는 것에 목표를 둔다. 실험 결과 본 논문에서 제안한 기법은 온칩 레지스터의 효과적인 활용을 통해 최대 41%에 가까운 가비지 컬렉션 비용을 감소시켰다.

Ⅱ. 관련 연구

현재 대부분의 상용 플래시 저장장치들은 로그 버퍼 기반의 FTL 기법을 사용하며, 일반적으로 BAST[2], FAST[3], SuperBlock[4] 기법들이 잘 알려져 있다. 이러한 로그 버퍼 기반 FTL은 플래시 내 블록을 로그 블록과 데이터 블록으로 나누고, 덮어쓰여지는 데이터를 로그 블록에 우선 기록한다. 로그 블록내의 빈 한 이 모두 사용된 경우 블록 합병이라는 과정을 통해서 로그 블록내의 기록한다. 로그데이터 블록에 기록한다. 이 과정에서 다수의 페이지 이주가 발생하기 때문에 이를 줄이기 위해 로그 버퍼 기반 FTL들은 상이한 데이터 블록과 로그 블록간의 연관 정책을 제안하고 있다. 여기서 연관 정책이란 특정 로그 블록에 얼마만큼의 데이터 블록이 연관될 수 있는 로그로 나타낸다.

BAST 기법의 경우 하나의 로그 블록은 하나의 데이터 블록과 연관될 수 있다. 즉, 하나의 로그 블록은 반드시 대응되는 하나의 데이터 블록의 데이터만을 저장할 수 있다. 반면 FAST 기법은 로그 블록의 공간 활용도를 높이기 위해 하나의 로그 블록이 모든 데이터 블록의 데이터를 저장할 수 있도록 한다. SuperBlock 기법은 특정 개수의 로그 블록의 집합이 동일한 개수의 데이터 블록의 집합에 대해서 연관되도록 제약을 둔다. 이는 FAST 기법과 같이 블록간의 연관도가 높은 경우 블록 합병 시 소요되는 시간이 매우 길어지기 때문이다. 참고로 이러한 데이터 및 로그 블록의 집합을 슈퍼 블록이라 부른다.

위에서 언급한 세 가지 FTL 기법들은 최대한 데이터 이주 횟수 감소에 초점을 두고 있으며, 본 논문과 달리 온칩 레지스터의 활용도를 높이기 위한 기법은 제공하지 않는다.

Ⅲ. 온칩 레지스터를 활용한 가비지 컬렉션 부하 감소 기법

1. 전체 구조

본 논문에서 제안하는 기법은 다음과 같이 크게 두 개의 세부 기법으로 이루어져 있다. 첫 번째 기법은 로그 블록 할당 정책으로 데이터 덮어쓰기에 따른 새로운 로그 블록 할당 시 수행된다. 본 기법은 데이터 블록내의 특정 페이지가 갱신되어 해당 데이터가 새로운 로그 블록에 기록할 때, 가급적이면 갱신된 데이터 블록과 동일한 플레인에 속한 블록을 대응되는 로그 블록으로 할당시키도록 한다. 두 번째 기법은 빈 블록 할당 정책으로 블록 합병 시 적용된다. 모든 FTL 기법은 블록 합병을 위해 임시적인 빈 블록의 할당이 필요하며, 이때 본 기법은 가급적이면 대상 로그 블록 혹은 데이터 블록과 동일한 플레인에 있는 빈 블록을 할당하도록 한다.

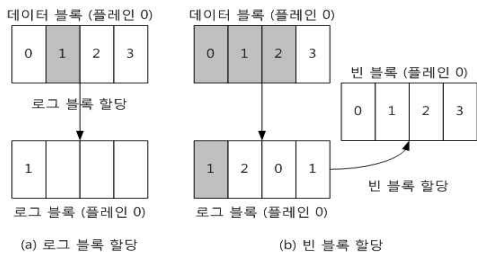

그림 1. BAST FTL 기법 적용

이미 언급하였듯이 각 FTL 기법들은 서로 상이

한 데이터 블록과 로그 블록간의 연관 방식을 사용한다. 이러한 FTL의 연관 방식에 따라 본 논문에서 제안한 기법들 역시도 상이하게 적용되어야 한다. 이어지는 장에서는 각 FTL 기법을 대상으로 본 논문에서 제안하는 온칩 레지스터 활용 기법이 어떻게 적용될 수 있는지를 세부적으로 설명한다.

2. BAST FTL 기법 적용

그림 1은 본 논문의 기법을 BAST FTL 기법에 적용할 때의 예를 보여준다. 그림1에서의 작은 사각형은 페이지를 나타내며, 큰 사각형은 블록을 보여준다. 여기서 블록은 총 네 개의 페이지로 구성되어 있다. 작은 블록내의 번호는 해당 페이지가 저장하고 있는 데이터의 논리주소를 보여준다. 회색의 사각형은 무효화된 데이터를 저장하고 있는 페이지를 나타내며, 반면 흰색의 사각형은 유효한 페이지를 나타낸다.

그림 1(a)는 로그 블록 할당 방식을 보여준다. BAST의 경우 특정 데이터 블록내의 페이지가 갱신될 경우 해당 페이지(예, 페이지 1)의 데이터를 대응되는 로그 블록에 기록한다. 본 기법은 덮어 쓰기 시 가능하면 해당 데이터 블록과 동일한 플레인에 있는 로그 블록이 할당되도록 시도 한다. 예를 들어 그림 1(a)의 경우 데이터 블록이 '플레인 0'에 포함되어 있었기 때문에, 로그 블록 역시 '플레인 0'에 속하도록 하였다.

그림 1(b)는 빈 블록 할당 정책의 예를 보여준다. 위의 그림에서와 같이 모든 로그 블록내의 빈 공간이 사용된 경우 FTL은 해당 로그 블록과 데이터 블록을 블록 합병하며, 이 과정에서 한 개의 빈 블록이 할당되어야 한다. 이후 로그 블록과 데이터 블록 내의 유효한 페이지는 해당 빈 블록에 복사되며, 로그 블록과 데이터 블록은 삭제된다. 마지막으로 빈 블록은 이전의 데이터 블록이 된다. 본 논문에서 제안한 기법은 온칩 레지스터의 활용을 최대화시키기 위하여 빈 블록 선택 시 데이터 블록 및 로그 블록과 동일한 플레인에 있는 블록을 선택하도록 시도한다. 그림 1(b)의 경우, '플레인 0'에 속한 빈 블록이 선택되었음을 알 수 있다. 이 경우 블록 합병에 따라 발생하는 모든 페이지의 이주가 온칩 레지스터의 활용만으로도 수행 가능해 진다.

3. FAST FTL 기법 적용

FAST 기법의 경우 하나의 로그 블록이 다수의 데이터 블록과 연관될 수 있다. 그림 2(a)는 로그 블록 할당시의 상황을 보여준다. '데이터 블록 0'에 덮어쓰기가 발생하여 해당 블록의 페이지가 '로그 블록 0'에 기록된 상태이다. 이후 '데이터 블록 1'에

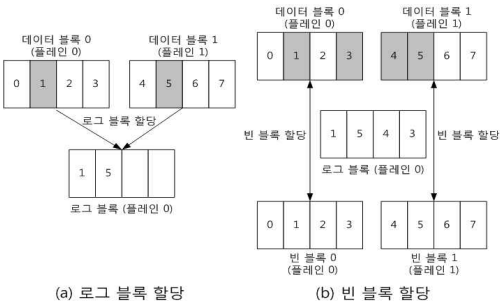

그림 2. FAST FTL 기법 적용

덮어쓰기가 발생하고 또한 '로그 블록 0'에 빈 페이지가 있기 때문에 해당 데이터는 '로그 블록 0'에 기록된다. 이때 '데이터 블록 0'과 '데이터 블록 1'은 같은 플레인에 속할 수도 속하지 않을 수도 있다. 본 예에서는 두 데이터 블록이 서로 다른 플레인에 속해있다고 가정하였다. 본 논문에서 제안한 기법은 로그 블록이 최초 할당될 때 기록되는 데이터의 데이터 블록과 동일한 플레인에 있는 블록이 로그 블록으로 선택되도록 한다.

그림 2(b)는 빈 블록 할당의 경우를 보여준다. FAST FTL의 경우 블록 합병을 위해 '데이터 블록 0'과 '데이터 블록 1'을 위해 두 개의 빈 블록을 선택해야 한다. 제안된 기법은 가급적 온칩 레지스터의 활용률을 높이기 위해 각 데이터 블록과 동일한 플레인에 있는 빈 블록을 할당시키도록 한다. 즉, '데이터 블록 0'을 위해서는 '플레인 0'에 있는 빈 블록을, 반면 '데이터 블록 1'을 위해서는 '플레인 1'에 있는 빈 블록을 할당한다. 본 예의 경우 블록 합병 시 발생하는 모든 페이지의 이주가 온칩 레지스터의 활용만으로 수행될 수 없다.

4. SuperBlock FTL 기법 적용

SuperBlock 기반의 FTL 기법은 BAST 기법과 FAST 기법이 혼합된 방식을 사용한다. 슈퍼 블록 내의 로그 블록과 데이터 블록 간에는 FAST와 같이 블록 연관 상의 제한을 두지 않으나, 슈퍼 블록끼리는 블록간의 연관이 불가능하다. 본 논문에서는 슈퍼블록에 새 로그 블록 할당 시 슈퍼 블록 내의 블록들과 동일한 플레인에 있는 로그 블록을 할당하도록 하며, 블록 합병을 위한 빈 블록 할당 시에도 슈퍼 블록 내의 블록과 동일한 플레인에 있는 빈 블록을 할당하도록 시도하였다.

Ⅳ. 실험 결과

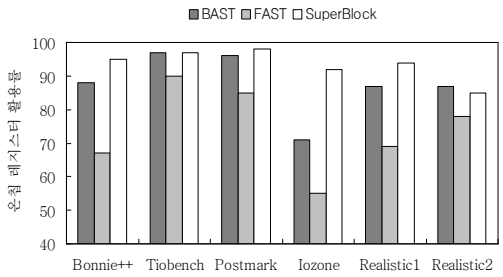

그림 3 온칩 레지스터 활용률(%)

1. 실험 환경

제안된 기법의 평가를 위해 본 논문에서는 플래시 저장장치용 기능 시뮬레이터를 사용하였다. 플래시 메모리 파라미터는 삼성 NAND 플래시 메모리 명세서[1]의 것을 사용하였다. 실험을 위해 다양한 저장장치 벤치마크(Bonnie++, Tiobench, Postmark, Iozone)와 실제 PC에서 다양한 응용을 수행하며 수집한 작업부하(Realistic1, Realistic2)가 사용되었다.

2. 실험 결과

그림 3은 BAST, FAST, SuperBlock 기법을 대상으로 다양한 벤치마크에서의 온칩 레지스터 활용률을 보여주고 있다. FAST의 경우 가장 낮은 활용률을 보였는데, 이는 FAST의 완전 연관 방식 때문이다. BAST에 비해 SuperBlock은 약간 더 우수한 활용률을 보여주었는데, 이는 슈퍼 블록내의 블록들을 동일한 플레인에 할당하는 방식이 BAST처럼 매번 할당하는 것 보다 유리했기 때문으로 해석된다.

표 1은 온칩 레지스터 활용 기법을 적용한 경우(with onchip regs)와 적용하지 않은 경우(wo onchip regs)의 성능을 비교하고 있다. 모든 FTL 기법과 벤치마크 프로그램에 대하여 30%~41%까지의 성능 개선이 발생함을 확인할 수 있다. FAST 기법의 경우에는 온칩 레지스터 활용률은 낮게 나왔으나, 페이지 이주 횟수를 많이 감소시켜 일부 벤치마크에서는 가장 우수한 성능을 제공하였다.

Ⅴ. 결 론

본 논문은 NAND 플래시 메모리의 온칩 레지스터를 활용한 가비지 컬렉션 부하 감소 기법을 제안하였다. 제안된 기법은 다양한 벤치마크 프로그램을 통해 평가 되었으며, 최대 41%의 가비지 컬렉션 부하가 감소됨을 확인하였다. 이는 많은 페이지 이주 연산이 온칩 레지스터만을 활용하여 수행 가능함을 보여준다. 실험 결과에서 확인할 수 있듯이 일부 벤치마크 프로그램에 대해서는 온칩 레지스터의 활용이 비교적 낮게 나왔다. 이는 빈 블록 선택 시 항상 원하는 플레인에 속한 블록을 얻지 못했기 때문이다. 따라서 이를 위한 개선된 빈 블록 관리 기법이 필요하다고 판단된다.

참 고 문 헌

[1] Samsung Corp. “K9WBG08U1M NAND Flash Memory,” 2007.

[2] J. Kim, J. Kim, S. H. Noh, S. L. Min, and Y. Cho. “A space-efficient flash translation layer for compact flash systems,” IEEE Trans. on Consumer Electronics, vol. 48, no. 2, pp. 366-375, 2002.

[3] S.W. Lee, D. J. Park, T. S. Chung,W. K. Choi, D. H. Lee, S.W. Park, and H. J. Song. “A log buffer based flash translation layer using fully associative sector translation,” ACM Trans. on Embedded Computing Systems, vol. 6, no. 3, 2007.

[4] J. Kang, H. Jo, J. Kim, and J. Lee. “A superblock-based flash translation layer for NAND flash memory,” in Proc. of International Conference on Embedded Software, pp. 161-170, 2006.

	BAST		FAST		SuperBlock	
	wo onchip regs	with onchip regs	wo onchip regs	with onchip regs	wo onchip regs	with onchip regs
Bonnie++	88.97	57.71	63.2	43.83	40.61	23.28
Tiobench	680.82	400.47	483.08	286.78	644.69	365.85
Postmark	3691.14	2153.67	2117.95	1300.31	3879.49	2176.62
Iozone	611.61	425.76	606.75	454.12	571.92	337.02
Realistic1	1161.61	734.27	921.75	631.26	1013.23	587.8
Realistic2	474.81	297.03	239.57	154.4	358.93	222.48

표 1. 온칩 레지스터 활용 유무에 따른 가비지 컬렉션 수행 시간 차이 비교 (단위: 초)