Journal of Digital Contents Society Vol. 16 No. 4 Aug. 2015(pp. 535-543)
http://dx.doi.org/10.9728/dcs.2015.16.4.535

플래시 메모리기반 저장장치에서 효율적 메타데이터 관리 기법

김동욱*, 강수용**

요 약

현재 NAND 플래시 메모리기반 저장장치는 NAND 플래시 메모리의 단점을 감추고 장점을 극대화해 나가며 그 활용 영역을 지속적으로 넓혀왔다. 특히, 이러한 저장장치는 NAND 플래시 메모리의 고유한 특성인 “쓰기 전 지우기” 특성을 감추기 위하여 내부적으로 플래시 변환 계층(Flash Translation Layer)이라 불리는 소프트웨어 계층을 포함하고 있다. 플래시 변환 계층은 호스트로부터 요청된 데이터를 관리하기 위한 메타데이터를 포함하며, 메타데이터는 호스트의 요청들을 처리하기 위해 자주 접근되는 데이터이므로 내부 메모리에 저장되어 관리된다. 따라서 메모리에 저장된 메타데이터는 전원손실이 발생하게 되는 경우 모두 소멸되므로, 메타데이터를 주기적으로 저장하고 초기화 과정을 통해 메타데이터를 메모리에 적재할 수 있는 메타데이터 관리 정책이 필요하다. 따라서 우리는 메타데이터 관리의 핵심 요구사항을 모두 만족하면서 효율적으로 동작하는 메타데이터 관리 정책을 제안하며, 실험을 통해 제안하는 기법의 효율성을 증명하였다.

키워드 : NAND 플래시 메모리, 메타데이터, 복구

Efficient Metadata Management Scheme in NAND Flash based Storage Device

Dongwook Kim*, Sooyong Kang**

Abstract

Recently, NAND flash based storage devices are being used as a storage device in various fields through hiding the limitations of NAND flash memory and maximizing the advantages. In particular, those storage devices contain a software layer called Flash Translation Layer(FTL) to hide the “erase-before-write” characteristics of NAND flash memory. FTL includes the metadata for managing the data requested from host. That metadata is stored in internal memory because metadata is very frequently accessed data for processing the requests from host. Thus, if the power-loss occurs, all data in memory is lost. So metadata management scheme is necessary to store the metadata periodically and to load the metadata in the initialization step. Therefore we proposed the scheme which satisfies the core requirements for metadata management and efficient operation. And we verified the efficiency of proposed scheme by experiments.

Keywords : NAND flash memory, Metadata, Recovery

1. 서론

현재 NAND 플래시 메모리는 1984년 처음 소개된 이후로 일상생활의 다양한 분야에서 널리 사용되고 있는 매체(medium)가 되었다. 이는 SD(Secure Digital) Card, eMMC(embedded

※ Corresponding Author: Sooyong Kang
Received : July 10, 2015
Revised : August 20, 2015
Accepted : August 24, 2015
* Department of Electronics and Computer Engineering, Hanyang University
** Division of Computer Science and Engineering, Hanyang University
Tel: +82-2-2220-1725,
email: sykang@hanyang.ac.kr

▣ 이 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2014R1A2A2A01004187).

Multi-Media Card), UFS(Universal Flash Storage), SSD(Solid State Drive)와 같은 NAND 플래시 메모리 기반의 저장장치들이 임베디드 시스템, 모바일 장치부터 개인용 컴퓨터와 고성능 컴퓨팅 시스템 분야까지 다양한 환경의 저장장치로 활용되고 있기 때문이다[1]. 이러한 NAND 플래시 메모리기반의 저장장치들은 일반적으로 NAND 플래시 메모리의 고유한 특성인 “ 쓰기 전 지우기 ”(erase-before-write) 문제를 해결하기 위해 Log-Structure File System(이하, LFS)과 같이 갱신(update)된 데이터를 새로운 영역에 저장하는 방법(out-place update)을 사용한다[2, 3]. 이러한 out-place update 기법은 가장 최신의 데이터 영역을 가리키기 위한 메타데이터와 새롭게 갱신되어 더 이상 유효하지 않는 데이터를 구분하기 위한 메타데이터가 필요하다[3].

이러한 메타데이터는 저장장치가 동작하는 동안 지속적으로 접근 및 변경되므로, 성능 향상을 위해 저장장치의 내부 메모리(예로, DRAM 혹은 SRAM)에 저장되어 값을 지속적으로 갱신한다[3, 4]. 그러므로 저장장치에 전원손실이 발생하게 되면 메모리에 저장된 메타데이터가 소멸 되는 문제가 항상 존재한다[5]. 따라서 저장장치의 초기화 단계에서 메모리에서 소멸된 메타데이터를 일관된(consistent) 상태의 값으로 초기화(혹은 복구)하는 과정이 반드시 필요하다. 이러한 메타데이터 초기화 과정은 저장장치에서 사용하는 메타데이터 관리 방법과 매우 밀접하게 연관되어 있으며, 메타데이터 관리 방법에 따라 서로 다른 성능을 보인다. 하지만 현재의 NAND 플래시 메모리기반 저장장치에서 사용하는 메타데이터 관리 기법과 각 기법의 장단점에 대해서는 널리 알려지지 않은 상황이다. 따라서 본 논문에서는 제한적으로 알려진 메타데이터 관리 기법과 각 기법의 장단점에 대해 알아보고, SSD와 같은 고성능, 대용량의 NAND 플래시 메모리기반 저장장치에서 사용할 수 있는 효율적인 메타데이터 관리 기법을 제안한다.

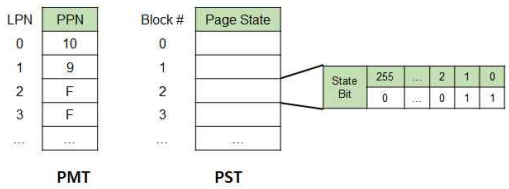
본 논문의 구성은 다음과 같다. 2장에서는 SSD에서 사용하는 메타데이터에 대해 간략히 소개한다. 3장에서는 관련된 메타데이터의 핵심 요구사항 및 기존 기법들에 대해 기술하고, 4장에서는 제안하는 부분 메타데이터 체크포인팅 기법을 소개한다. 4장에서는 실험을 통해 제안하는 기법의 성능을 평가하며, 끝으로 5장에서 이 논문의 결론을 맺는다.

2. 배경지식

2.1 SSD의 메타데이터(Metadata)

페이지 단위의 매핑을 사용하는 NAND 플래시 메모리기반 저장장치에서 최신의 데이터 영역을 가리키는 메타데이터를 일반적으로 페이지 매핑 테이블(Page Mapping Table, PMT)이라 하며, 저장장치에서 더 이상 유효하지 않은 데이터를 구분하기 위한 메타데이터를 페이지 상태 테이블(Page State Table, PST)이라 한다. 일반적으로 페이지 매핑 테이블과 페이지 상태 비트맵은 페이지 매핑을 사용하는 NAND 플래시 메모리기반 저장장치에 가장 기본이 되는 메타데이터로 알려져 있다[3].

(그림 1) 메타데이터 구조



(Figure 1) Metadata Structure

이러한 목적을 위해 사용하는 메타데이터의 구조는 (그림 1)과 같다. PMT의 각 엔트리에는 해당 논리 페이지가 저장된 물리 페이지의 주소인 물리 페이지 번호(Physical Page Number, PPN)가 저장된다. PST는 각 물리 페이지의 상태 정보가 저장된다. (그림 1)의 예에서 페이지 상태 테이블(PST)의 값은 유효하지 않은(invalid) 상태의 경우는 1로, 유효한(valid) 상태의 값은 0으로 표현된다. 만약 각 블록의 모든 페이지들이 유효하지 않은 상태인 경우, 블록 삭제(block erase) 과정에서 페이지의 복사 연산 없이 바로 삭제될 수 있다. 반대로 유효한 상태의 페이지가 존재하는 경우, 블록 삭제 과정에서 페이지를 복사해야 하는 추가적인 과정이 발생하며 이로 인해 성능이 하락하게 된다.

3. 메타데이터 관리 기법

일반적으로 LFS와 같은 쓰기 방법을 사용하는 시스템의 메타데이터 관리 기법은 크게 역매핑(Reverse Mapping, RM), 메타데이터 체크포인팅 (Metadata Checkpointing, MC) 혹은 스냅샷(snapshot), 메타데이터 로깅(Metadata Logging, ML) 으로 구분할 수 있으며[2, 6, 7, 8], 이는 NAND 플래시 메모리기반 저장장치에서도 동일하게 사용되는 기법들이다. 본 장은 각 기법의 자세한 설명과 장단점에 대해서 기술하며, 이를 통해 메타데이터 관리의 핵심 요구사항을 도출한다. 또한 도출된 결과를 통해, 각 기법이 이를 얼마나 만족시키는 지에 대해서 기술한다.

3.1 역매핑(Reverse Mapping, RM) 기법

역매핑 기법은 페이지의 예비 영역(Spare Area)에 저장된 역매핑 정보를 이용하여 메타데이터를 재생성(rebuild)하는 초기화 기법을 사용한다. 일반적으로 역매핑 기법을 사용하기 위해서 물리 페이지의 예비 영역(spare area)에 저장되는 역매핑 정보는 논리 페이지 번호와 타임스탬프 값이며, 이러한 역매핑 정보를 통해 메타데이터를 가장 최신의 상태로 항상 초기화 할 수 있다.

따라서 역매핑 기법은 메타데이터의 일관성을 보장하기 위한 추가적인 공간과 추가 쓰기가 전혀 필요하지 않으며, 급작스러운 전원손실(Sudden Power-Off)과 같은 시스템 결함과 관계없이 항상 메타데이터의 일관성을 완벽하게 보장할 수 있다. 하지만 역매핑 기법은 초기화 과정에서 메타데이터를 재생성하기 위해 저장장치의 모든 데이터 영역을 스캔(scan)해야 하므로, 초기화 시간이 저장장치의 용량에 따라 선형적으로 증가한다.

3.2 메타데이터 체크포인팅(Metadata Checkpointing, MC) 기법

메타데이터 체크포인팅 기법은 저장장치의 특정한 영역(메타데이터 영역)에 저장된 메타데이터를 메모리에 적재하는 초기화 기법을 사용한다. 이러한 초기화 기법을 사용하기 위해 메타데이터 체크포인팅 기법은 주기적으로 메모리에 있는 메타데이터를 특정한 영역(메타데이터 영역)에 저장한다.

메타데이터 체크포인팅 기법은 초기화 단계에서 메타데이터 영역에 저장된 메타데이터만을 스캔하여 메모리에 적재하면 되기 때문에, 역매핑 기법보다 빠른 초기화 시간을 보장한다. 하지만 저장장치에서 사용하는 메타데이터의 크기가 큰 경우에는 여전히 저장장치의 빠른 초기화 시간을 보장하기 어렵다. 또한 메타데이터 체크포인팅 기법은 시스템 결함이 발생한 경우에 메타데이터 영역에 저장되지 못하고 소멸된 메타데이터가 존재할 수 있으므로 메타데이터의 일관성을 완벽하게 보장하지 못하며, 메타데이터를 저장하기 위한 추가적인 쓰기와 공간(메타데이터 영역)이 필요하다. 메타데이터를 저장하는 추가적인 쓰기는 호스트로의 요청 처리 시간을 지연시켜 저장장치의 성능을 감소시킨다. 특히, 이러한 성능 감소는 메타데이터의 크기가 클수록, 메타데이터 저장의 주기가 짧을수록 더욱 심해진다.

3.3 메타데이터 로깅(Metadata Logging, ML) 기법

메타데이터 로깅(Metadata Logging, ML) 기법은 메타데이터의 변화 정보인 로그(Log)를 통해 메타데이터를 재생성하는 초기화 기법을 사용한다. 이러한 초기화 기법을 사용하기 위해 메타데이터 로깅 기법은 메타데이터의 변화 정보를 저장장치의 특정한 영역(로그 영역)에 저장하며, 로그의 저장으로 인해 추가되는 NAND 페이지의 쓰기 양을 최소화하기 위해서는 로그를 페이지 크기 단위로 모아서 저장한다.

메타데이터 로깅 기법은 로그 영역에 저장된 로그의 양에 따라 초기화 시간이 결정된다. 다시 말해 저장장치에 저장된 데이터의 양이 적을수록, 저장장치의 초기화 시간이 빨라진다. 또한 메타데이터 체크포인팅 기법과 달리 메타데이터의 변화 정보만을 저장하기 때문에, 메타데이터의 저장으로 인해 발생하는 추가 쓰기로 인한 저장장치의 성능 감소는 크지 않으며, 무시할만한 수준이다. 하지만 메타데이터 로깅 기법은 시스템 결함이 발생하게 되면 메모리에서 모으던 로그들이 로그 영역에 저장되지 못하고 소멸될 수 있으므로 메타데이터의 일관성을 완벽하게

보장하지 못하며, 로그를 저장하기 위한 추가적인 공간이 필요하다. 또한 저장장치에 저장된 데이터의 양이 일정 수준을 증가하게 되면, 오히려 메타데이터 체크포인팅 기법보다 초기화 시간이 더욱 증가할 수 있다. 왜냐하면 메타데이터 체크포인팅 기법은 순차적으로 메타데이터가 저장되기 때문에 실제 메타데이터 값만이 저장될 수 있지만, 메타데이터 로깅 기법은 워크로드(workload)의 패턴에 따라 임의로 메타데이터 값이 저장되기 때문에 각각의 값을 구분하기 위한 추가 정보가 필요하기 때문이다. 예로, 메타데이터 체크포인팅 기법은 PMT에 저장되는 실제 값인 물리페이지 번호(PPN)만을 저장하면 되지만, 메타데이터 로깅 기법은 PPN의 엔트리 값인 논리페이지 번호(LPN)도 같이 저장해야한다.

3.4 메타데이터 관리의 핵심 요구사항

일반적으로 저장장치에서 메타데이터를 관리하기 위해 필요한 핵심적인 요구사항은 각 메타데이터의 관리 기법들[2, 6, 7, 8]을 제안한 이유를 통해 쉽게 유추할 수 있으며, 우리는 이러한 핵심적인 요구사항을 **1) 일관성: 메타데이터의 일관성 보장, 2) 속도: 메타데이터의 빠른 초기화 시간 보장, 3) 효율성: 메타데이터 관리를 위한 추가적인 쓰기의 최소화**로 정의하였다. 이러한 구분에 따라 앞에서 기술한 각 메타데이터 관리 기법들이 이를 얼마나 만족시키는지에 대해 비교해 보았으며, 그 결과는 <표 1>과 같다.

<표 1> 메타데이터 관리를 위한 핵심 요구사항

	Consistency	Speed	Efficiency
RM	○ (○)	× (×)	○ (○)
MC	○ (×)	△ (×)	× (×)
ML	○ (×)	△ (×)	△ (×)

( ) : After system crash
○ : Sufficiency, △ : Depending on conditions, × : Insufficiency

<Table 1> Requirements for Metadata Management

3.5 메타데이터의 일관성 보장

메타데이터 관리의 세 가지 요구사항 중 가장 중요한 것은 메타데이터의 일관성 보장이다. 왜냐하면 저장장치의 일관성이 보장되지 않는다면 중요한 시스템의 저장장치로서 사용될 수 없기 때문이다. 일반적으로 상용 SSD는 이러한 문제를 해결하기 위해 슈퍼커패시터(supercapacitor)와 전원손실 보호(power-loss protection) 기법을 통해 메모리에 저장된 데이터의 손실을 방지한다[9]. 그러나 저장장치에 사용하는 슈퍼커패시터의 용량이 증가할수록 제품의 가격도 증가하게 되므로 적절한 용량의 슈퍼커패시터를 사용하는 것이 중요하다.

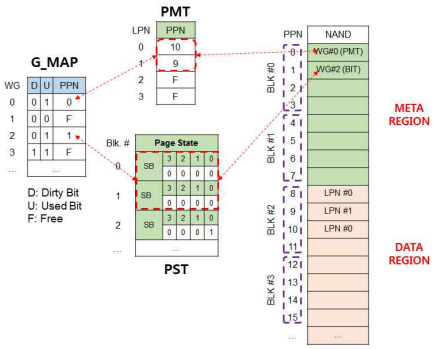
4. 부분 메타데이터 체크포인팅

본 장은 메타데이터 관리의 핵심 요구사항을 언제나 만족시킬 수 있는 새로운 메타데이터 관리 기법인 부분 메타데이터 체크포인팅 (Partial Metadata Checkpointing, 이하 PMC) 기법을 기술한다. PMC 기법의 핵심 주안점들은 다음과 같다. 1) 메타데이터의 일관성 보장, 2) 메타데이터 저장으로 인한 저장장치의 성능 저하 최소화, 3) 메타데이터의 빠른 초기화 시간 보장이다.

4.1 내부메모리 메타데이터와 저장영역의 구조

PMC는 메타데이터 저장으로 인한 저장장치의 성능 감소를 최소화하기 위해 메타데이터 전체 부분을 저장하는 것이 아니라, 실제 변화가 발생한 부분만을 저장한다. 메타데이터를 부분적으로 저장하기 위해 PMC가 사용하는 메타데이터와 NAND 플래시 메모리 영역의 구조는 (그림 2)와 같다.

(그림 2) 내부메모리 메타데이터



(Figure 2) In-memory Metadata

PMC는 전체 NAND 플래시 메모리 영역을 META, DATA 영역으로 구분한다. META 영역은 PMC가 사용하는 메타데이터가 저장되는 영역이고, DATA 영역은 호스트로부터 요청된 데이터가 저장되는 영역이다.

PMC는 일반적인 NAND 플래시 메모리기반 저장장치에서 사용되는 메타데이터(PMT, PST) NAND 플래시 메모리의 읽기/쓰기 단위인 페이지 크기의 배수로 구분하여 관리하며, 각 관리 단위를 Write Granularity(이하 WG)라 부른다. (그림 2)는 PMT와 PST가 WG 단위로 구분되어 관리되는 구조를 보여준다. (그림 2)의 G_MAP은 PMT와 PST의 각 WG들이 저장된 물리 페이지의 위치가 저장되는 메타데이터이다. G_MAP에는 각 WG가 저장된 위치 이외에도 D/U 비트가 존재한다. D는 더티(Dirty) 비트로 각 WG에 해당하는 메타데이터가 변경이 되어 META 영역에 저장된 값과 다른 경우 1로 설정(set)되며, 메타데이터 영역에 안전하게 저장되어 META 영역에 저장된 값과 같은 경우에 0으로 설정된다. U는 이용(Used) 비트로 각 WG에 해당하는 메타데이터의 값이 존재하는 경우 1로 설정되며, 해당 WG 영역에 쓰기가 발생하지 않았거나 트림(TRIM) 커맨드를 통해 WG 영역의 모든 메타데이터 값이 초기화 되는 경우 0으로 설정된다. D/U 비트는 G_MAP의 PPN 값이 사용하는 전체 크기 중 2비트를 사용한다. 예로, G_MAP에서 PPN 값으로 4바이트(32비트)를 사용한다면, 그 중 2비트는 D/U 비트로 사용되며 나머지 30비트는 PPN 비트로 사용된다. 이 경우에 30비트의 PPN 값은 4KB 물리 페이지 크기를 기준으로 약 2TB의 주소 영역을 표현할 수 있으므로, META 영역을 가리키기에는 충분한 크기이다.

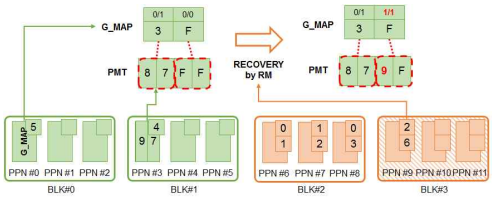
4.2 메타데이터 체크포인팅과 초기화

PMC에서 메타데이터 체크포인팅 과정은 더티 WG 플러싱(flushing)과 G_MAP을 저장하는 과정으로 구분된다. 이러한 메타데이터 체크포인팅 과정은 메타데이터를 저장으로 인해 발생하는 저장장치의 부하(overhead)를 최소화하기 위해 주기적으로 발생한다. 이러한 주기는 실제 저장해야하는 메타데이터의 크기(즉, 더티 WG의 양)와 초기화 과정에서 스캔해야하는 페이지의 수가 결정하며, 사용하는 값에 따라 메타데이터 저장의 효율성이 결정된다.

메타데이터 초기화 과정은 체크포인팅의 역순으로 진행된다. 먼저, 최신의 체크포인팅 과정에서 저장된 G_MAP을 META 영역으로부터 메모리에 적재한다. 그 후, G_MAP의 사용(U) 비트가 1인 WG를 메모리에 모두 적재한다. 이는 전체 메타데이터 중 실제 사용되는 영역의 메타데이터만을 스캔하여 적재하므로, 전체 메타데이터를 스캔하지 않아도 되는 장점을 갖는다. 그러나 모든 메타데이터 영역이 사용되는 경우에는 메타데이터 스캔의 감소효과를 얻을 수 없게 된다.

PMC의 메타데이터 체크포인팅 과정은 특정한 주기적로 발생하기 때문에 메타데이터 체크포인팅 간격 사이에 급작스러운 전원 손실이 발생하게 되면, META 영역에 저장되지 못하고 소멸되는 메타데이터가 존재한다. 이러한 메타데이터의 손실을 막기 위해, PMC는 메타데이터 체크포인팅 주기 사이에 저장되는 블록들의 정보를 저장하는 기법을 사용하며, 이러한 정보는 슈퍼커패시터에 의해 보호된다. 따라서 초기화 단계에서 메모리에서 소멸된 메타데이터가 존재하는 경우, 해당 블록들의 페이지에 저장된 역매핑 정보를 통해 메타데이터를 복구하게 된다. 이러한 초기화 방법에 따른 복구 예제는 (그림 3)과 같다.

(그림 3) PMC 기법의 복구 과정



(Figure 3) Recovery process in PMC

4.3 PMC의 기여도

본 논문을 통해 새롭게 제안한 PMC 기법은 다음과 같이 메타데이터 관리의 세 가지 요구사항을 만족시킨다. 하나, 메타데이터 체크포인팅과 데이터 블록의 페이지에 저장된 역매핑 정보를 통해 메타데이터의 일관성을 항상 보장한다. 둘, 실제 변화된 메타데이터만을 저장(store)하므로 메타데이터 저장으로 인한 저장장치의 성능

저하를 최소화한다. 셋째, 실제 사용 중인 메타데이터만을 초기화 단계에 복구하게 되므로 메타데이터의 빠른 초기화 시간을 보장한다.

5. 실험 결과

5.1 시뮬레이터 설정

본 논문을 통해 제안하는 메타데이터 관리 기법인 PMC의 성능을 검증하기 위해, 본 실험은 트레이스 구동(trace-driven) 시뮬레이터를 통해 성능을 측정하였으며, 시뮬레이터의 설정과 사용한 워크로드의 특징은 <표 2>, <표 3>와 같다.

<표 2> 시뮬레이터 설정

Configuration	Value
SSD capacity	512GB
Channel x Way	6 x 4
Over-Provisioning (OP)	64 GB
NAND page size	4KB
Pages per block	256
PMT/BIT size	512MB / (about) 18MB
WG size	variable
G_MAP size	variable

<Table 2> Simulator configuration

<표 3> 워크로드 특징

	Total write/read (GB)	Avg. write/read (sectors)	Running time	NoE: Num. of Entry
LINUX	5.96 / 0.2	383 / 38	497 sec	1,415,168
TPC-C	20.84 / 1.93	33 / 47	9,062 sec	622,256
WEB	0.89 / 2.71	30 / 31	7,355 sec	567,572

<Table 3> Workload characteristics

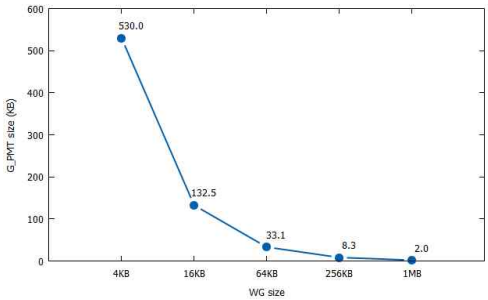
본 실험은 대용량의 저장장치에서도 효율적으로 동작하는 메타데이터 관리 정책을 목표로 하므로, SSD의 용량을 512GB로 설정하였다. 저장장치의 용량이 512GB인 경우, 페이지 매핑 기반의 SSD에서 사용하는 메타데이터의 크기는 약 530MB가 된다. 이는 메타데이터의 소멸을 막기 위해 메타데이터 전체를 저장(store)/적재(load)하는 것만으로도 매우 큰 오버헤드를 갖게 됨을 의미한다.

<표 3>에서 Linux는 우분투 리눅스를 설치한 워크로드이고, TPC-C는 TPC-C 벤치마크를 10 웨어하우스(warehouse)의 설정으로 실행하여 추출한 워크로드이며, WEB은 윈도우즈7 운영체제에서 웹서핑, 파일 다운로드와 같은 일반적인 사용자의 작업들을 추출한 워크로드이다. 또한 <표 3>의 NoE(Number of Entry)는 각 워크로드의 시뮬레이션 동작을 완료한 뒤, 각각의 읽기/쓰기 요청들을 처리하기 위해 접근된 모든 PMT와 PST의 엔트리 수를 의미한다(PST의 각 엔트리는 비트(bit) 단위이기 때문에 블록 단위로 추출하였다). 이러한 NoE와 전체 읽기/쓰기 양을 통해 각 워크로드가 저장장치의 영역 접근 패턴을 유추할 수 있다. 예로 TPC-C 워크로드는 쓰기 양에 비해 NoE가 매우 작으므로 로컬리티(locality)가 매우 높은 것을 알 수 있으며, 반대로 WEB과 LINUX 워크로드는 로컬리티가 TPC-C에 비해 낮음을 알 수 있다.

5.2 WG 크기에 따른 성능 변화

우리는 WG 크기에 따른 PMC의 성능 변화를 측정하기 전에, WG의 크기에 따른 G_MAP의 크기 변화를 측정하였다. WG의 크기는 페이지 크기(4KB)부터 블록크기(1MB)까지에 변화시켰으며, 그 결과는 (표 4)와 같다.

(그림 4) WG 크기에 따른 G_MAP의 크기



(Figure 4) G_MAP size based on WG size

(그림 4)를 통해 알 수 있듯이 WG의 크기가 증가할수록 G_MAP의 크기가 감소하며, 이는 WG의 크기를 증가시킬수록 G_MAP의 저장으로 인한 쓰기 양을 줄일 수 있음을 의미한다. 하지만 WG의 크기가 증가하면 할수록, G_MAP의 메타데이터 쓰기 양도 같이 증가한다. 왜냐하면 각 WG가 포함하는 메타데이터의 일부분만이 변화된 경우에도, 메타데이터의 저장은 WG 단위로 동작하므로 실제 변화된 부분보다 많은 부분

의 읽기/쓰기가 발생할 수 있기 때문이다. 따라서 WG의 크기에 따른 G_MAP의 더티/사용 상태의 개수 변화를 측정하였으며 그 결과는 <표 4>와 같다.

<표 4> 더티/사용 상태의 엔트리 개수

	4KB	16KB	64KB	256KB	1MB
LINUX	1,551/1,550	448/448	148/148	80/80	65/65
TPC-C	1,152/718	405/220	154/84	68/38	46/24
WEB	2,824/1253	1,147/678	401/300	131/112	44/39
Tot. Entry	135,681	33,921	8,481	2,121	531

<Table 4> Number of dirty/used state entry

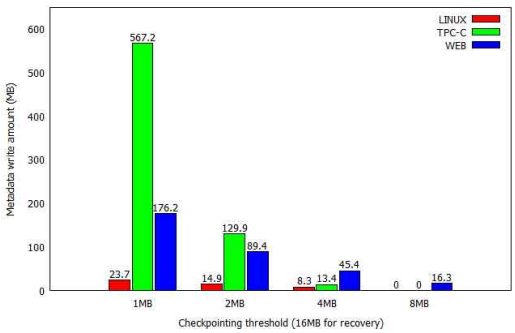
<표 4>를 통해 알 수 있는 것은 WG의 값이 감소할수록 초기화 단계에서 스캔해야하는 메타데이터 영역의 크기와 WG 플러싱을 위한 쓰기 양이 감소한다는 것이다. 하지만, 앞 절에서 본 것같이 WG의 크기가 작아질수록 G_MAP의 크기가 증가하기 때문에 효율적인 값을 사용해야 한다.

5.3 WG 플러싱 임계값에 따른 성능 변화

5.2장의 실험 결과를 바탕으로 우리는 WG의 크기를 16KB로 설정하였고, 이때의 G_MAP 크기는 약 132.5KB이다. 이러한 크기는 병렬 구조를 가지는 SSD에서 큰 부하(overhead)없이 처리될 수 있는 크기이다. 그리고 <표 3>의 결과는 메타데이터 체크포인팅이 발생하면 더티 WG 개수만큼의 메타데이터를 한 번에 저장해야하는 부하가 발생하게 됨을 의미하며, 이러한 더티 WG가 저장되지 못하고 소멸된 경우에는 역매핑 정보를 통해 초기화 과정에서 역매핑 정보를 스캔하여 복구해야 하는 페이지가 증가하게 된다. 따라서 PMC는 더티 WG의 개수가 특정한 임계값(threshold)를 초과하거나, 복구를 위해 스캔해야 하는 페이지가 일정 수를 초과하는 경우에 메타데이터 체크포인팅을 발생시킨다. 이러한 임계값을 효율적으로 선정하기 위해, 다양한 값에 따른 메타데이터 쓰기 양을 측정하였으며, 그 결과는 (그림 5)와 같다.

실험에 사용된 모든 워크로드는 접근 범위가 넓지 않기 때문에 일정 크기 이상을 임계값으로 사용하면, 메타데이터 쓰기 양을 크게 감소시킬 수 있다. 따라서 본 논문에서는 체크포인팅을 위한 더티 WG의 최대 크기를 4MB로 설정하였으며, 이는 체크포인팅 과정에 최대 4MB의 쓰기가 발생할 수 있음을 의미한다. 또한 초기화 단계에서 메타데이터 복구를 위해 읽어야 하는 페이지의 최대 크기를 16MB로 설정하였다.

(그림 5) 임계값에 따른 메타데이터 쓰기 양



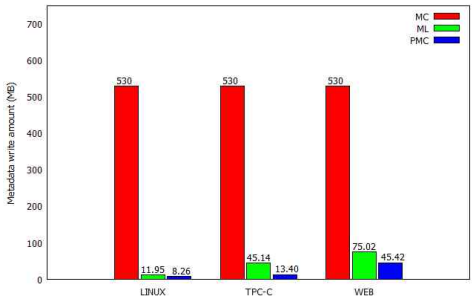
(Figure 5) Metadata write amount based on threshold

5.4 PMC의 성능 비교

위의 결과를 통해 제안하는 PMC는 적절한 임계값과 WG 크기를 설정한 경우 메타데이터 체크포인팅의 큰 오버헤드 없이 동작함을 알 수 있다. 이러한 결과를 기준으로, SSD에서 일반적으로 많이 사용하는 메타데이터 관리 정책인 메타데이터 체크포인팅(MC), 메타데이터 로깅(ML) 기법과의 성능을 비교하였다. 단, RM 기법은 초기화 단계에서 항상 예비 영역에 저장된 복구 정보를 통해 메타데이터를 재생성하기 때문에 추가적으로 메타데이터를 저장하는 과정이 필요하지 않으므로, 메타데이터 쓰기 양은 항상 0이다.

먼저 각 기법들이 LINUX, TPC-C, WEB 워크로드를 실행하는 동안 발생하는 메타데이터의 쓰기 양을 측정하였다. MC는 한 번의 메타데이터의 쓰기가 발생한 경우로 가정하였다. ML은 4KB의 물리 페이지에 511개의 8바이트 크기의 (LPN, PPN) 페어가 저장되며, 남은 8바이트에는 현재 페이지에 저장된 로그의 개수와 타임스탬프를 저장된다고 가정하였다.

(그림 6) 메타데이터 쓰기 양 비교 (RM=0)



(Figure 6) Comparison of metadata write amount (RM = 0)

(그림 6)의 결과를 통해 PMC는 ML보다 메타데이터 쓰기의 양을 감소시킬 수 있음을 알 수 있다. 이러한 경향은 워크로드의 접근 범위가 좁을수록 더욱 크게 나타난다(TPC-C, WEB, LINUX의 순).

6. 결론 및 향후연구

본 논문에서는 메타데이터 관리의 세 가지 요구사항을 만족시키면서 메타데이터 관리로 인한 성능 감소는 크지 않은 부분 메타데이터 체크포인팅(PMC) 기법을 제안하였다. 제안하는 기법은 WG 크기 단위로 메타데이터를 구분하여 관리하며, 메타데이터 저장 및 초기화 단계에서 실제 사용하는 영역만을 적재/저장하여 메타데이터 관리로 인한 오버헤드를 감소시킨다. 제안한 PMC는 이러한 설계 목적에 따라 동작할 때, 기존에 사용되던 메타데이터 관리 기법보다 메타데이터 쓰기 양을 크게 감소시킬 수 있음을 실험을 통해 검증하였다. 우리는 향후연구를 통해 PMC를 실제 저장장치 시스템에 구현하여, 기법의 취약점을 발견 및 보완하는 연구를 진행해 나갈 것이다.

References

[1] H. Jung, S. Kang, and J. Cha, “An Asymmetric Buffer Management Policy for SSD,” Journal of digital contents society, vol. 12, no. 2, pp. 141-150, 2011

[2] E. Gal and S. Toledo, “Algorithms and Data Structures for Flash Memories,” ACM Computing Survey, vol. 37, no. 2, 2005.

[3] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy, “Design tradeoffs for SSD performance,” Proc. USENIX ATC ‘08 , pp. 57-70, Jun 2008.

[4] S. Park, S. Kang, “Considerations for Designing an Integrated Write Buffer Management Scheme for NAND-based Solid State Drives”, Journal of digital contents society, vol. 14, no. 2, pp. 215-222, 2014.

[5] M. Zheng, J. Tucek, F. Qin, and M. Lillibridge, “Understanding the Robustness of SSDs under Power Fault,” Proc. USENIX FAST ‘13, pp. 271-284, 2013

[6] K. Sun, S. Baek, J. Choi, D. Lee, S. H. Noh, and S. L. Min, “LTFTL: Lightweight time-shift flash translation layer for flash memory based embedded storage,” Proc. EMSOFT ’08, pp. 51-58, Oct, 2008.

[7] C. Wu, T. Kuo, and L. Chang, “The Design of efficient initialization and crash recovery for log-based file systems over Flash memory,” ACM Transaction on Storage, Vol 2, No. 4, pp. 449 - 467, Nov 2006.

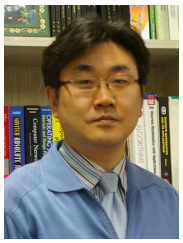
[8] K. S. Yim, J. Kim, and K. Koh, “A fast start-up technique for flash memory based computing systems,” Proc. ACM SAC ‘05, pp. 843-849, 2005

[9] “Enhanced Power-Loss Data Protection in the Intel Solid-State Drive 320 Series,” http://www.intel.co.kr/content/dam/www/public/us/en/documents/technology-briefs/ssd-320-series-power-loss-data-protection-brief.pdf.

김 동 욱

2009년 : 호서대학교 컴퓨터공학부 (학사)
2011년 : 한양대학교 전자컴퓨터통신공학과 (석사)

현 재 : 한양대학교 전자컴퓨터통신공학과 (박사과정)
관심분야 : 플래시메모리 기반 저장 시스템, 운영체제, 임베디드 시스템



강 수 용

1996년 : 서울대학교 수학과 (학사)
1998년 : 서울대학교 전산과학과 (석사)
2002년 : 서울대학교 전기컴퓨터공학부 (박사)

2003년~현 재: 한양대학교 컴퓨터공학부 교수
관심분야 : 운영체제, 멀티미디어시스템, 스토리지 시스템, 플래시 메모리, 차세대 메모리, 분산 컴퓨팅