MySQL/InnoDB DWB특성을 고려한 FTL 기술

박현우 ○ 이상원
성균관대학교 소프트웨어학과
music2eye.hw@gmail.com swlee@skku.edu

An FTL Technique for MySQL InnoDB's DWB

Hyun-Woo Park○ Sang-Won Lee
College of Software, Sungkyunkwan University

요 약

플래시메모리 SSD는 플래시메모리의 erase-before-write 특성으로 인해 플래시변환계층(FTL) 모듈을 이용하여 기존 하드디스크와 같은 블록 디바이스처럼 사용할 수 있다. 본 논문에서는 대표적인 page mapping FTL을 기반으로, MySQL/InnoDB의 DWB 영역에서 발생하는 순차적이고 순환적인 쓰기 특성을 이용하여, DWB 영역과 나머지 영역의 쓰기 스트림을 분리한 기능을 제안한다. 이 기법을 통해 의미있는 성능 향상, copyback 횟수 및 수명 개선을 이룰 수 있음을 보인다.

1. 서 론

FTL을 연구할 때 많은 사람들이 SSD시뮬레이터를 이용하여 성능테스트를 하는데, 실제 SSD와 시뮬레이터 간의 차이가 발생하기 때문에 시뮬레이터에서 좋은 결과를 냈다 하더라도 실제 SSD에서는 그리 좋은 성능을 내지 못하는 경우가 발생하기도 한다. 실제 SSD와 비슷한 테스트 결과를 얻기 위해서는 시뮬레이터가 아닌 실제 SSD를 사용해야 하는데, OpenSSD의 경우 RAM사이즈나 SSD 사이즈 등을 쉽게 조절 가능하며, FTL을 직접 수정 및 개발이 가능하기 때문에 연구용으로 사용하기에 적합하다. 따라서 본 논문의 연구에서는 OpenSSD를 사용하여 구현 및 실험을 진행하였다.

기존의 SSD들은 page mapping 방식을 기반의 FTL을 주로 사용한다. 쓰기 요청은 크게 랜덤한 쓰기와 순차적인 쓰기 두 가지로 분류된다. page mapping FTL은 이 중 랜덤한 쓰기에 효율적이다. 그러나, 두 종류의 요청이 섞여서 들어올 경우 그렇지 않은 경우에 비해 많은 copyback이 발생하게 되고, SSD의 성능 및 수명에 영향을 미치게 된다.

본 논문에서는 현재 많이 쓰이고 있는 DBMS인 MySQL의 쓰기 요청의 특성에 따라 다른 방식으로, DWB(Double Write Buffer)와 나머지 쓰기 요청을 각각 별개의 공간에서 관리함으로써 Copyback을 줄이고 이에 따른 SSD의 성능과 수명 향상을 이끌어 내었다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 배경지식을 소개한다. 3장에서는 쓰기 특성에 따른 FTL 최적화를, 4장에서는 성능 평가 및 분석, 5장에서는 결론 및 향후 연구로 본 논문을 마무리한다.

2. 배경지식

2.1 OpenSSD

OpenSSD는 크게 Host Interface Layer(HIL), Flash Translation Layer(FTL), Flash Interface Layer(FIL)로 구성되어 있다.

HIL은 SATA 호스트 명령과 버퍼 관리는 담당하는 계층으로, 호스트로부터 SATA컨트롤러에 IO요청이 전달되면, 해당 명령을 SATA 이벤트 큐에 삽입한다. 그 후 FTL이 순차적으로 요청을 처리하도록 돕는다.

FTL은 플래시 메모리를 하드디스크와 같은 블록 디바이스로 인식할 수 있도록 하는 소프트웨어 계층이다. FTL은 주소 매핑, 웨어 레벨링, 가비지 콜렉션 등을 담당하며, IO 요청의 처리는 FTL이 플래시 메모리에 해당 연산을 전달함으로써 이루어진다. FTL은 성능 및 안정성을 높이기 위해 여러 기법들이 존재하는데, Jasmine 펌웨어에는 Tutorial FTL, Greedy FTL, Dummy FTL이 구현되어 있다.

FIL은 플래시 메모리를 담당하며, FTL로부터 전달된 플래시 명령의 실제 동작은 LLD(Low-level device driver)에 의해 수행되며, 정상동작 과정에서 발생한 예외 상황은 인터럽트 컨트롤러에 의해 탐지된 후, FTL이 핸들링한다.

OpenSSD 펌웨어는 소스코드가 C언어로 되어 있어 이용하기가 수월하며, 개발 또한 용이하다. OpenSSD 펌웨어 헤더파일 및 소스파일들은 그림1과 같이 구성되어 있다.

2.2 FTL

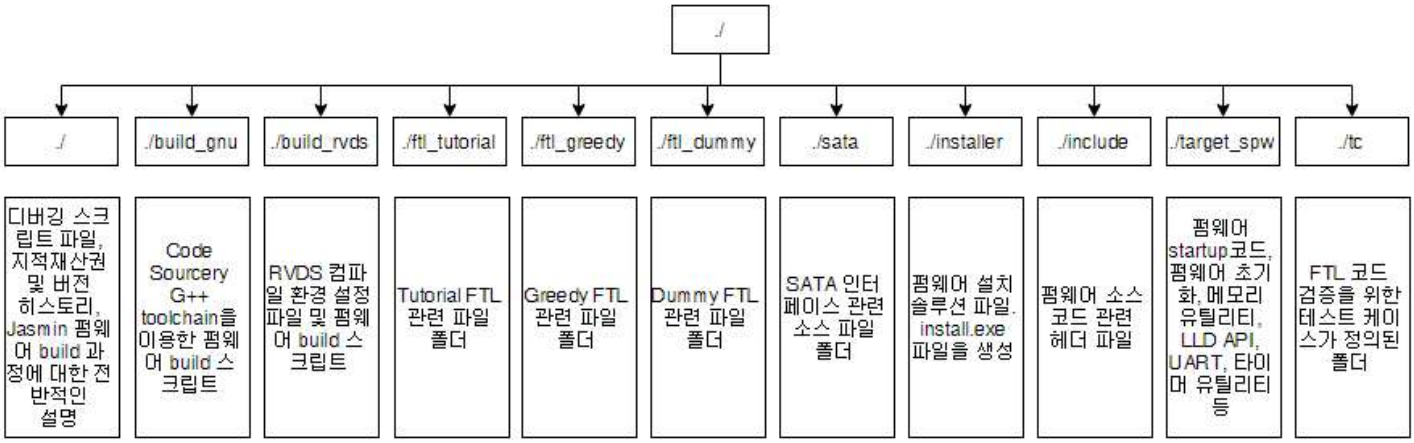

그림 1 OpenSSD 펌웨어 구조

SSD는 비휘발성, 빠른 접근속도, 낮은 전력소모 등 HDD에 비해 많은 장점을 가짐에도 불구하고 NAND 플래시 메모리의 erase-before-write 특성 때문에 HDD를 대체하기가 간단하지 않다. 일단 데이터가 NAND 플래시 메모리에 기록되는 경우 원래의 데이터를 포함하는 영역이 소거될 때까지 해당 영역을 덮어쓸 수 없다.

NAND 플래시 메모리는 Flash Translation Layer(FTL)라 불리는 소프트웨어 레이어를 사용함으로써 호스트 시스템에서 불리한 특성을 감추고, 기존의 블록 디바이스처럼 사용할 수 있게 해준다.

FTL의 역할은 크게 논리적 주소와 실제 주소 차이에 따른 올바른 주소를 반환해주는 논리-물리 주소 매핑, 특정 소자가 반복되어 프로그래밍되는 상황을 방지하고 고르게 프로그래밍 되도록 해주는 웨어 레벨링(wear leveling), earse-before-write 특성에 따른 사용하지 않지만 이미 쓰여져 있는 불필요한 블록을 사용할 수 있도록 해주는 가비지 컬렉션 3가지가 있다. 이 중 논리-물리 주소 변환과 가비지 컬렉션을 어떻게 하느냐에 따라 FTL의 성능이 크게 달라진다.

2.3 DWB(Double Write Buffer)

MySQL의 InnoDB의 기본 page size는 16kbyte이고, OS의 기본 I/O 단위는 4kbyte이다. 이로 인해 특정 페이지의 일부분만 I/O 작업이 완료되고, 다른 부분은 아직 완료되지 않은 상태에서 system failure가 발생할 경우 page가 깨질 수 있는데 이를 방지하고 복구하는 데 DWB가 사용된다.

InnoDB buffer pool에서 변경된 page의 내용이 disk로 내려 써지는 경우, 그 내용을 DWB에 기록한다. 이때 기록하는 내용은 해당되는 페이지 정보와 operation 정보가 기록된다. 즉, flush 이벤트가 발생하면, 먼저 DWB에 해당 내용을 기록하고 그 다음에 대상 페이지의 내용을 변경하는 작업을 진행하게 된다.

이러한 DWB는 읽기 요청이 거의 발생하지 않으며, 쓰기 요청은 순차적이라는 특징이 있다.

본 논문에서는 이러한 DWB의 특성을 고려하여 이를 다른 쓰기요청과 분리하여 매핑 테이블을 관리하도록 하였다.

3. 쓰기 특성에 따른 FTL 최적화

먼저 OpenSSD로 DWB 쓰기 요청이라는 정보를 전달하기 위해 kernel 3.18버전과 MySQL 5.7.2버전의 코드를 수정하였다.

OpenSSD의 greedy FTL을 기반으로 Hybrid FTL 코드를 작성하였다. 기존의 page mapping 테이블과 별개로 DWB를 위한 위한 매핑 테이블과 valid page 수를 기록하기 위한 vcount 테이블을 생성하였다. 그리고 DWB 영역을 관리하기 위한 garbage collection 함수, vpn 관리 함수, vcount 관리 함수, write 함수 등을 새롭게 만들었다.

OpenSSD가 쓰기 요청을 받으면 해당 요청이 DWB 쓰기 요청인지 아닌지 판단하고, DWB 쓰기 요청일 경우 추가적으로 구현한 영역으로, 아닐 경우 기존의 page mapping 영역으로 보내서 관리한다.

이로 인해 랜덤한 쓰기 요청과 순차적인 쓰기 요청이 섞여서 많은 copyback을 발생시키는 문제를 해결하였고, 성능 향상 및 Copyback감소로 인한 WAF값 감소와 SSD의 수명 향상을 이끌어내었다.

4. 성능 평가 및 분석

4.1 실험환경

[표 1] 실험 환경

OS	Ubuntu 14.04 LTS
CPU	Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
RAM	64GB
MySQL	5.7.2
Benchmark tool	Linkbench
Benchmark option	Transactions : 5100000개 # Operation mix for request phase

	# numbers are percentages and must sum to 100
	addlink = 29
	deletelink = 10
	updatelink = 25
	countlink = 0
	getlink = 0
	getlinklist = 0
	getnode = 0
	addnode = 9
	updatenode = 23.5
	deletenode = 3.5

4.2 실험결과

[표 2] 실험 결과

FTL	Read	Write	GC	Copyback	Ops
Original FTL	2385657	9998645	117957	5208561	171
Custom FTL	2385315	9995041	111828	4447374	182

[표 3] Custom FTL에서의 DWB와 전체의 실험평가 항목 비교

	Read	Write	GC	Copyback
DWB	0	1146595	8982	0
전체	2385315	9995041	111828	4447374

4.3 분석

같은 양의 작업을 수행했을 때 original FTL과 custom FTL을 비교하기 위해 같은 시간이 아닌 같은 트랜잭션 수를 수행하였을 때의 결과를 비교하였다.

표2 에서 기존의 FTL과 비교하였을 때, Ops는 171에서 182로 6.43% 증가하였다. 이로 보아 FTL에서 추가적인 작업으로 인한 Overhead보다 이로 인한 성능 향상이 더 많음을 알 수 있다. 또한 표3 에서 보이는 바와 같이 DWB쓰기 요청을 따로 분리하여 관리할 경우 DWB영역에서 불필요한 Copyback이 발생하지 않음을 알 수 있다. 그로 인해 표2 에서 전체 Copyback 횟수가 14.61%감소하고 G.C 횟수가 5.20% 감소했음을 알 수 있다.

5. 결론 및 향후 연구

본 논문에서는 MySQL InnoDB의 DWB 쓰기 특성에 따른 FTL 최적화로 성능 향상 및 Copyback 횟수의 감소와 수명 향상을 이루었다.

향후 연구로는 특정 어플리케이션에만 적용할 수 있는 것이 아닌 범용적으로 이를 적용할 수 있는 방안에 대하여 연구할 계획이다.

사사

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 SW컴퓨팅산업원천기술개발사업(SW스타랩)의 연구결과로 수행되었음" (IITP-2015-0-00314)

참 고 문 헌

[1] OpenSSD Jasmine board FTL Developer's Guide, http://www.openssd-project.org/mediawiki/images/Jasmine_FTL_Dev_Guide_v.1.2.pdf

[2] OpenSSD Jasmine board Technical Reference Manual, http://www.openssd-project.org/mediawiki/images/Jasmine_Tech_Ref_Manual_v.1.4.pdf