논문 2010-47CI-5-5

RAM 디스크를 이용한 FTL 성능 분석 시뮬레이터 개발

(Development of Simulator using RAM Disk for FTL Performance Analysis)

임 동 혁*, 박 성 모*

(Dong-Hyuk Ihm and Seong-Mo Park)

요 약

NAND 플래시 메모리는 기존의 HDD 보다 빠른 접근 속도, 저전력 소비, 진동에 대한 내성 등의 이점을 바탕으로 PDA를 비롯한 여러 모바일 장치부터, 임베디드 시스템, PC에 이르기까지 사용 영역이 넓어지고 있다. DiskSim을 비롯한 HDD 시뮬레이터들이 다양하게 개발되어 왔으며, 이를 바탕으로 소프트웨어 또는 하드웨어에 대한 개선점을 찾아냄으로써 유용하게 사용되었다. 하지만 NAND 플래시 메모리나, SSD에 대해서는 리눅스 기반의 몇 개의 시뮬레이터만이 개발되었으며, 실제 스토리지 장치나 PC등이 사용되는 운영체제가 윈도우즈인 것을 고려하면 윈도우즈 기반의 NAND Flash 시뮬레이터가 꼭 필요하다고 볼 수 있다. 본 논문에서 개발한 NAND Flash FTL 성능 분석을 위한 시뮬레이터인 NFSim은 윈도우즈 운영체제에서 구동되는 시뮬레이터로, NAND 플래시 메모리 모델 및 FTL 알고리즘들은 각각 윈도우즈 드라이버 모델 및 클래스로 제작되어 확장성이 용이하고, 각 알고리즘의 성능을 측정한 데이터는 그래프를 통해 표시되므로, 별도의 툴을 사용할 필요가 없다.

Abstract

NAND flash memory has been widely used than traditional HDD in PDA and other mobile devices, embedded systems, PC because of faster access speed, low power consumption, vibration resistance and other benefits. DiskSim and other HDD simulators has been developed that for find improvements for the software or hardware. But there is a few Linux-based simulators for NAND flash memory and SSD. There is necessary for Windows-based NAND flash simulator because storage devices and PC using Windows. This paper describe for development of simulator-NFSim for FTL performance analysis in NAND flash. NFSim is used to measure performance of various FTL algorithms and FTL wear-level. NAND flash memory model and FTL algorithm developed using Windows Driver Model and class for scalability. There is no need for another tools because NFSim using graph tool for data measure of FTL performance.

Keywords: NAND, RAM DISK, FTL, SIMULATOR, SSD

I. 서 론

플래시 메모리는 하드 디스크 드라이브에 비해서 전력 소비량이 낮으며, 무게가 가볍고, 충격에 강한 장점 때문에 PDA 같은 임베디드 시스템의 저장 장치로 주로 사용된다. 최근에는 이러한 휴대용 정보 기기들이 널리 보급됨으로써, 플래시 메모리[1]도 지속적으로 발전하고 있는 추세이다. 플래시 메모리에는 NOR 타입과 NAND 타입이 있으며, NAND 타입의 플래시 메모리는 NOR 타입에 비해 집적도가 뛰어나 용량이 크고, 쓰기 연산과 삭제 연산의 속도가 빠른 특징을 가지기 때문에 데이터 저장 장치로 주로 사용된다. NAND 플래시가 여러 장점을 보유하면서도 하드 디스크 드라이브를 대체하지 못하는 이유는 가격이 높고, 읽기 속도보다 쓰기 속도가 약 5배 느리다는 점, 가비지 컬렉션 동안 시스템이 느려지는 현상, 하드 디스크 드라이브에 비해 수명이 짧은 점 등이 단점으로 작용하기 때문이다. 또

* 정회원, 전남대학교 전자컴퓨터공학과
(Dept. of Electronics and Computer Engineering. Chonnam National University)
※ 본 연구에 사용된 tool은 IT-SoC 사업단 및 IDEC의 지원을 받았습니다.
접수일자: 2010년6월3일, 수정완료일: 2010년8월31일

한 덮어쓰기가 불가능하여 갱신 연산을 수행할 때 추가적인 삭제 연산이 발생하는 단점이 있으며, NAND 플래시 메모리 기반의 장치는 성능을 향상시키기 위해서 이러한 제약을 극복해야 한다. 이에 관련한 여러 가지 방법들이 연구되었으며, 그 중 FTL (Flash Translation Layer)[2] 측면에는 효과적인 주소 변환, 가비지 컬렉션, wear-leveling에 관련한 여러 테크닉이 있으며, 이는 NAND 플래시 메모리의 성능을 향상 시킬 뿐만 아니라 마모도를 줄여 플래시 메모리의 수명을 연장하는 역할을 한다[3]. 결국 효과적인 FTL 알고리즘을 개발하는 것이 NAND 플래시를 이용하는 시스템에 매우 중요한 역할을 하며, FTL 알고리즘 개발에 활용할 수 있는 자료가 필요하지만, 이에 관련한 시뮬레이터는 그 수가 극히 적으며, 대다수가 리눅스를 기반으로 하고 있다. 실제 NAND 플래시를 이용하는 사용자의 운영체제가 윈도우즈인 것을 고려해 볼 때 알고리즘 개발을 위한 자료로 이용하는 파일 I/O 생성 및 분석, 시뮬레이션 등이 윈도우즈 환경에서 처리되는 것이 더 정확한 데이터를 획득할 수 있고, 더욱 효과적인 알고리즘을 개발하는데 기초자료로 사용할 수 있을 것이다. 본 논문에서는 이러한 점에 착안하여 NAND 플래시 메모리의 FTL의 성능 분석 및 평가를 할 수 있는, 윈도우즈 환경에서 구동되는 시뮬레이터의 설계 및 구현을 하였다.

II. FTL 성능 분석 시뮬레이터 개발 내용

1. 하드웨어 모델링 컴포넌트 설계 및 구현

가. CFlash 클래스 설계 및 구현

플래시 메모리의 특성을 임의로 변화시키면서 실험할 수 있는 환경을 제공하기 위해서 전체 NAND 플래시 메모리 용량 및 블록 크기, 페이지 크기, 연산의 수행 시간 등을 제어 변수로 조절할 수 있도록 설계하였다. 따라서 시뮬레이션의 결과를 기존의 플래시 메모리 뿐만 아니라 새로운 플래시 메모리에 적용되는 FTL 모듈의 설계 방향을 제시 하는 등의 자료로 이용할 수 있다. 또한 플래시 메모리에 대한 연산 기능을 제공할 뿐만 아니라 연산의 수행 시간 검증 기능, 인자 검증 기능, 연산 적합성 검사 기능을 제공하여 결함 발생 방지 기능 및 결함 처리가 제대로 처리 되는지를 확인할 수 있다. CFlash 클래스는 그림 1과 같이 인터페이스 모듈과 연산을 담당하는 모듈로 나뉜다.

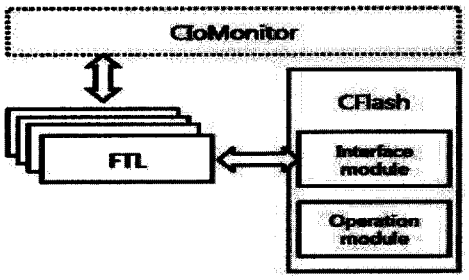


그림 1. CFlash 모듈 구성
Fig. 1. CFlash Module Configuration.

연산 모듈은 인터페이스 모듈과 밀접하게 연관되어 작동을 하며, 인자 적합성 검사, NOP 검사, 마모도 검사를 수행한다. 인자 적합성 검사는 인터페이스 모듈에 주어진 인자의 값이 플래시 메모리의 구성에 적합한지 검사를 한다. 예를 들면, Read 함수를 호출하였을 경우 적절한 주소 값이 제공되었는지, 버퍼의 값이 올바른 주소 값인지의 여부 등을 검사하는 역할을 한다. NOP 검사에서는 각 페이지에 덮어쓰기가 가능한 횟수를 위반하여 쓰기 연산이 수행되는지를 검사한다. 마모도 검사는 각 블록에 대해서 삭제 연산이 가능한 횟수가 제한되어 있기 때문에 플래시 메모리의 모든 블록에 대해서 마모도를 기록하고, 이 값이 제한된 값을 초과하는지를 검사한다. 연산 수행 모듈과 수행 시간 검사 모듈은 동시에 수행되는 모듈로써, 각 인터페이스의 정의에 따라 플래시 메모리 연산을 수행하고, 각 연산이 정확한 수행 시간을 준수할 수 있도록 모니터링 한다. 수행 시간 검사 모듈은 각 모듈이 실행되는 동안에 측정된다.

나. RamDisk 디바이스 드라이버 설계 및 구현

본 논문에서는 대용량 NAND 플래시 메모리를 모델링하기 위해서 WDF(Windows Driver Foundation)를 이용한 RAM 디스크 드라이버를 구현하였다. 이 방식은 하드 디스크 드라이브를 이용하여 파일 I/O를 생성하는 시스템보다 처리 속도가 높고, NAND 플래시 메모리의 용량이나 수량 조절을 용이하도록 해서 하드웨어 구성이 손쉬운 장점이 있다[4]. 그림 2는 RAM 디스크를 이용한 시뮬레이션을 간략히 나타낸 것이다. 유저 모드에서 실행되는 어플리케이션인 시뮬레이터에서 논리주소로 쓰기 연산을 실행하면, NTDLL.DLL을 통해서 커널 모드로 진입하고, 파일 시스템 드라이버로 이 요청이 전달된다. 이 요청은 다시 I/O 관리자에게 전달되고, IO 관리자는 이 I/O 요

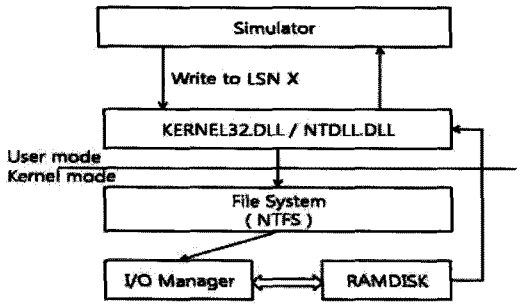


그림 2. RAM 디스크를 이용한 시뮬레이션
Fig. 2. Simulation using a RAM disk.

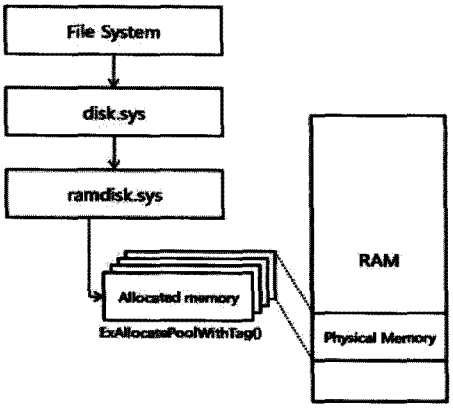


그림 3. RAM 디스크 동작
Fig. 3. RAM disk process.

청의 최종 목적지를 고려하여 IRP(Interrupt Request Packet)을 생성하고, IRP 스택에 각 계층이 수행하여야 할 정보를 저장한다. RAM 디스크 드라이버는 이 IRP를 전달 받은 후 관련 된 작업을 하고, IRP에 논리 주소와 길이, 자료, 지시자 등의 정보를 담아 요청했던 시뮬레이터에게 반환한다. 시뮬레이터는 이러한 정보를 바탕으로 실제 NAND 플래시 메모리에 데이터를 쓴 것과 같은 동작을 수행한다.

이 RAM 디스크 드라이버는 하드 드라이브를 대신하는 역할을 하는 가상 디스크로, 그림 3과 같이 물리 메모리에서 ExAllocatePoolWithTag 함수를 이용하여 페이지 아웃 되지 않는 메모리를 할당하고, 이를 디스크 드라이브처럼 사용할 수 있도록 한다. 또, 파일 시스템에서 I/O 가 발생하면 해당하는 IRP(Interrupt Request Packet)에서 논리 주소와 사이즈를 필터링 하는 필터 드라이버의 역할을 겸한다. 하드 디스크 드라이브를 이용하여 I/O를 발생시켜 패킷을 필터링 하는 방식보다 RAM 디스크 드라이브를 이용한 방식은 파일 I/O를 처리하는 속도가 매우 빠른 장점이 있다.

2. 소프트웨어 컴포넌트 설계 및 구현

가. COperation 클래스 설계 및 구현

COperation 클래스는 그림 4와 같이 동작하며, RAM 디스크 드라이버에 의해 생성된 드라이브에 패킷 생성 방식, 또는 Full-range 방식으로 I/O 동작을 발생 시켜 해당 연산에 대한 결과를 I/O Controller가 획득하여 CIoMonitor로 전송하는 역할을 한다. RAM 디스크 드라이버에 의해 생성된 드라이브는 윈도우즈에서 일반 디스크 드라이브로 인식되므로, 별도의 인터페이스가 필요 없이 Win32 API를 통해 접근 가능하다. 패킷 생성 방식은 사진, 문서, 음악 파일 등 비교적 저용량인 데이터를 전송하는 상황을 가정하여 그 결과를 확인하기 위해, 패킷 컨트롤러에서 8K 바이트에서 4M 바이트의 패킷을 임의로 생성하여 처리하도록 하였다. Full-range 방식은 이와 반대로 멀티미디어 파일 등 대용량 파일을 장치로 전송할 경우 발생하는 상황을 가정하고, 그 결과를 확인할 수 있도록 할당된 NAND 플래시 메모리 크기로 I/O가 발생하도록 처리하였다.

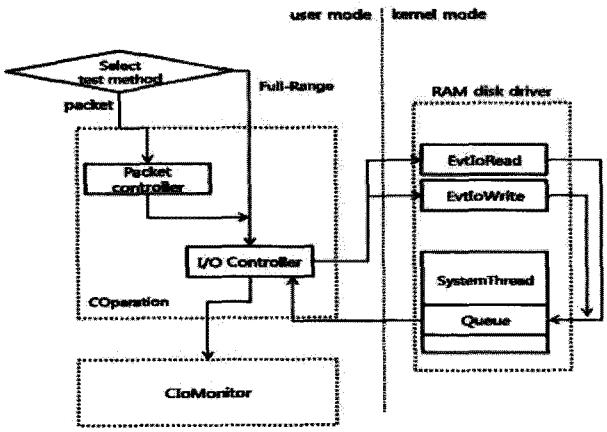


그림 4. COperation 클래스 동작
Fig. 4. COperation process.

나. CIoMonitor 클래스 설계 및 구현

CIoMonitor 클래스는 그림 5와 같이 COperation 클래스 동작 후, RAM 디스크에서 발생한 결과 값을 인자로 하여, 각각의 FTL 모듈 및 CFlash 모듈에 인터페이스를 제공하여 메인 스레드에서 이들을 구동시키는 역할을 한다. 또한 CIoMonitor 클래스는 각 FTL 모듈 동작 후 응답시간을 비롯한 결과 값을 CGraph 모듈로 전송한다. CIoMonitor 클래스는 각각의 FTL에 직관적이고, 일관된 인터페이스를 제공함으로써, FTL 모듈의 추가가 용이하고, 여러 가지 알고리즘의 성능 평가가 더욱 쉬워지도록 설계하였다.

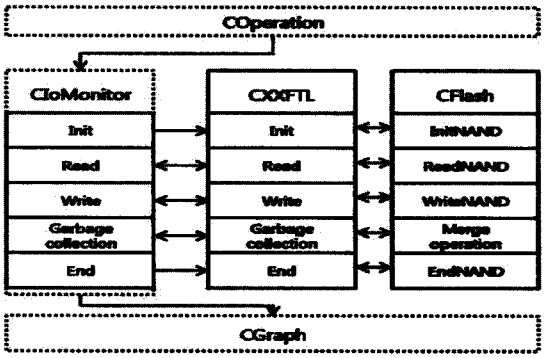


그림 5. CIoMonitor 동작
Fig. 5. CIoMonitor process.

다. CGraph 클래스 설계 및 구현

CGraph 클래스는 CIoMonitor에서 전송된 각 FTL 알고리즘의 속도 등을 그림 6과 같이 그래프로 표시하여 명확하게 알아 볼 수 있도록 하는 기능을 제공함과 동시에, XML을 이용하여 결과 값을 차트로 저장하는 기능을 제공하도록 설계하고 구현하였다. 본 논문에서 구현한 시뮬레이터는 기존의 시뮬레이터와 다르게 이러한 GUI 모듈들을 제공함으로써, 별도의 툴에서 결과 값을 분석하거나 그래프를 표시할 필요가 없다.

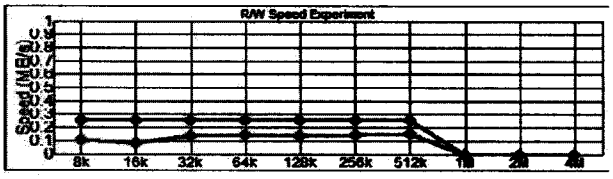


그림 6. CGraph에서 제공하는 그래프 표시 기능
Fig. 6. CGraph provide graph display function.

3. FTL 성능 측정 시뮬레이터 전체 시스템 구조

그림 7은 시뮬레이터의 전체 시스템 동작을 나타낸 것이다. 시스템 동작을 자세히 살펴보면, 먼저 COperation의 동작을 위해 테스트 방식을 지정하고, 이에 따라 패킷 방식이면 패킷 컨트롤러에 의해 패킷 사이즈가 지정되고, full-range 방식이면 곧 바로 I/O Controller가 제어한다. I/O Controller에서는 Win32API를 통해 RAM 드라이브에 접근하고, NTDLL.DLL을 통해 커널로 진입, 파일 시스템과 I/O 매니저에 각 연산에 맞는 제어 신호를 전송하고, I/O 매니저는 이에 맞는 IRP를 생성하여 RAM 디스크 드라이버에서 이에 대한 처리를 한다. I/O 루틴 완료 후에는 드라이버 내에 생성되어 있는 시스템 스레드에서 제어하고 있는 큐에 이 결과 값들이 삽입되고, 차례가 되면 COperation의 I/O

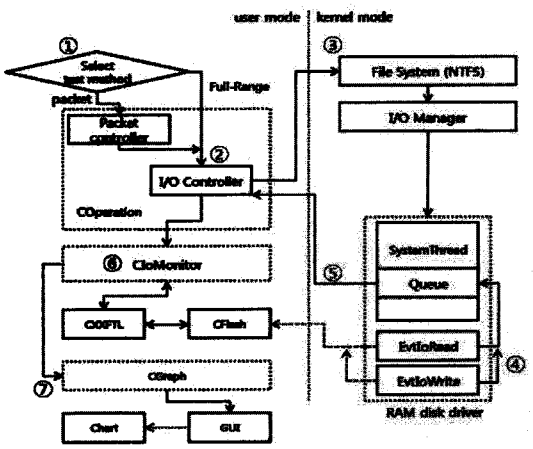


그림 7. 전체 시스템 동작
Fig. 7. Integral System Architecture.

Controller에 반환된다. 반환된 이 결과 값들은 CIoMonitor로 전송되어 각 FTL 모듈 및 CFlash에서 사용되고, 알고리즘 성능 분석 후 결과는 CGraph 모듈로 전송되어 그래프로 표시되거나, 사용자의 선택에 따라 차트로 저장된다.

Ⅲ. FTL 성능 측정 실험 및 결과

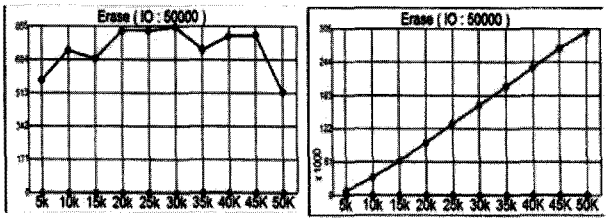
본 논문에서는 FTL 알고리즘의 I/O 동작 속도를 측정하기 위해서 Packet 생성 방식을 사용하였다. 페이지 수준의 사상 방식을 사용하는 FTL[5]과 블록 수준의 사상 방식을 사용하는 FTL[6], 그리고 FAST[7] 알고리즘을 사용하여 각각의 결과를 비교하였으며, 실험은 실제 사용자가 파일을 전송하는 것과 마찬가지로 파일의 생성, 읽기, 쓰기의 과정을 거치며, 신뢰성 있는 결과를 획득하기 위하여 각각 50000회씩 실험을 수행하였다.

1. Packet 생성 방식에 따른 속도 측정

Packet 생성 방식은 8K 바이트의 데이터부터, 4M 바이트의 데이터를 할당하여 이에 대한 삭제 연산 횟수를 측정하며, 이미지 파일, 문서, MP3 파일의 이동을 가정하고, 각 FTL 알고리즘에 따라 성능 차이가 있음을 확인하기 위해 수행하였다. Packet 생성 방식의 삭제 연산 횟수를 측정한 결과는 도식화하여 표현하였다.

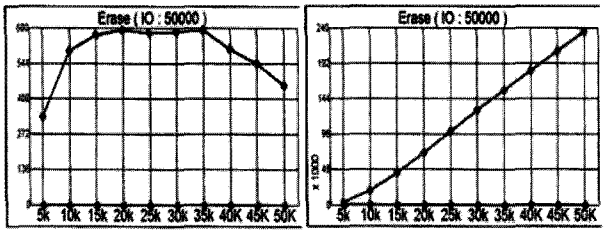
가. 3가지 FTL들의 성능 평가

실험 방식은 NAND 플래시 메모리에 Packet 생성 방식으로 50000회씩 수행하였다. 그림 8은 Packet 생성 방식으로 블록 사상 방식 FTL의 삭제 연산 횟수를 측



I/O Count	Erase Count	Erase Sum
5K	571.00	9827.00
10K	744.00	34845.00
15K	721.00	65162.00
20K	847.00	98394.00
25K	849.00	132668.00
30K	866.00	166956.00
35K	749.00	201429.00
40K	812.00	235776.00
45K	821.00	271914.00
50K	524.00	305648.00

그림 8. 블록 사상 방식 FTL의 삭제 연산 측정 결과
Fig. 8. Result of Block mapped FTL erase operation measurement.

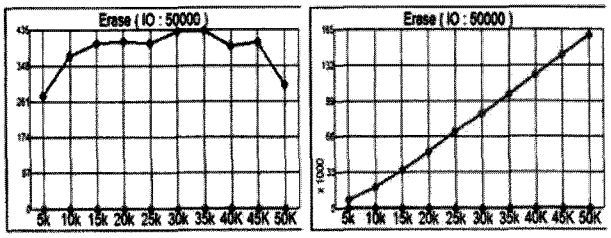


I/O Count	Erase Count	Erase Sum
5K	327.00	3757.00
10K	612.00	20918.00
15K	667.00	45023.00
20K	634.00	72492.00
25K	678.00	101211.00
30K	692.00	129201.00
35K	681.00	157377.00
40K	612.00	183715.00
45K	527.00	209855.00
50K	455.00	236224.00

그림 9. 페이지 사상 방식 FTL의 삭제 연산 측정 결과
Fig. 9. Result of Block mapped FTL erase operation measurement.

정한 것으로, 각 I/O 횟수 별 삭제 연산의 횟수와 삭제 연산 횟수의 누적 값을 나타내었다. 블록 사상 방식의 FTL은 3가지 방식 중 가장 삭제 연산의 횟수가 많았으며, 이는 성능 하락의 주요 원인이 된다.

그림 9는 페이지 사상 방식 FTL의 삭제 연산 횟수를 측정한 것을 나타낸 것으로, 블록 사상 방식 FTL과 비교했을 때, 삭제 연산의 수가 약 22% 감소한 것을 알 수 있다. 일반적으로 페이지 사상 방식은 블록 사상 방식에 비해 삭제 연산의 횟수가 적기 때문에 성능이 빠르지만, 사상 테이블을 저장하는 RAM 사용량이 블록 사상 방식 보다 크다. 이 실험에서는 RAM 사용량은 고려하지 않고, 각 FTL의 삭제 연산 횟수만을 비교하였다.



I/O Count	Erase Count	Erase Sum
5K	271.00	7663.00
10K	374.00	20819.00
15K	399.00	36612.00
20K	414.00	53627.00
25K	420.00	71172.00
30K	427.00	88620.00
35K	444.00	106721.00
40K	395.00	124642.00
45K	416.00	142953.00
50K	319.00	160723.00

그림 10. FAST FTL의 삭제 연산 측정 결과
Fig. 10. Result of FAST FTL erase operation measurement.

그림 10은 FAST FTL의 삭제 연산 측정 결과를 나타낸 것으로, 페이지 사상 방식 보다 약 31%, 블록 사상 방식보다 약 47% 가량 삭제 연산의 횟수가 줄어들었음을 알 수 있다.

Ⅳ. 결 론

NAND 플래시는 일반적으로 윈도우즈 OS를 사용하는 PC에서 가장 많이 사용하는데, 지금까지는 리눅스 기반에서 NAND 플래시 FTL의 성능을 측정하였다. 본 논문에서는 윈도우즈 기반에서 구동되는 NAND 플래시 메모리의 FTL에 대한 성능 평가 도구를 개발하여 FTL 개발에 사용할 수 있는 좀 더 정확한 자료를 획득할 수 있도록 하였다. 또한 테스트를 위한 디스크 입출력 패턴을 생성하고, 이를 이용하는 부분을 하드에서 처리하지 않고, 램 디스크에서 모두 처리하도록 하여 그 속도를 비약적으로 향상 시켰다. NAND 플래시의 동작 중 삭제 연산의 횟수가 성능 향상에 가장 큰 영향을 미치며, 본 논문에서 구현한 시뮬레이터는 여러 방식으로 제작된 FTL 알고리즘에 대해서 삭제 연산 횟수를 측정하고, 이에 따른 성능 평가를 수행함으로써, FTL 알고리즘 성능 개선 작업의 기초 자료로 사용할 수 있도록 하였다. 또한 여러 하드웨어 구성을 지원함으로써, 미 출시된 NAND 플래시 및 SSD의 FTL 알고리즘을 구현하고 개선하기 위해 이용할 수 있을 것이다.

참 고 문 헌

[1] AGRAWAL, N., PRABHAKARAN, V., WOBBER, T., DAVIS, J. D., MANASSE, M. S., AND PANIGRAHY, R. "Design tradeoffs for ssd performance." In Proceedings of the USENIX Annual Technical Conference, pp. 57 - 70, June 2008.

[2] E. GAL AND S. TOLEDO. "Algorithms and Data Structures for Flash Memories," ACM Computing Survey 37, 2, 138 - 163, June 2005

[3] W. G. Jeon and Y. S. Cho, "An equalization technique for OFDM and MC-CDMA in a multipath fading channels," in Proc. of IEEE Conf. on Acoustics, Speech and Signal Processing, pp. 2529-2532, Munich, Germany, May 1997.

[3] JUNG, D., CHAE, Y., JO, H., KIM, J., AND LEE, J. "A Group-based Wear-Leveling Algorithm for Large-Capacity Flash Memory Storage Systems." In Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES), pp. 160 - 164. 17, September 2007.

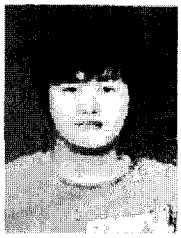
[4] GANGER, G., WORTHINGTON, B., AND PATT, Y. "The DiskSim Simulation Environment Version 3.0 Reference Manual."

[5] GUPTA, A., KIM, Y., AND URGAONKAR, B. "DFTL: A Flash Translation Layer Employing Demand-based Selective Caching of Page level AddressMappings". In Proceedings of the International Conference on Architectural Support for Programming Languages and Operating System (ASPLOS), pp. 229 - 240, March 2009.

[6] KANG, J., JO, H., KIM, J., AND LEE, J. "A Superblock-based Flash Translation Layer for NAND Flash Memory." In Proceedings of the International Conference on Embedded Software (EMSOFT), pp. 161 - 170, October 2006.

[7] LEE, S., PARK, D., CHUNG, T., LEE, D., PARK, S., AND SONG, H. "A Log Buffer based Flash Translation Layer Using Fully Associative Sector Translation". IEEE Transactions on Embedded Computing Systems 6, 3, 18, 2007.

저 자 소 개



임 동 혁(학생회원)-교신저자
2008년 전남대학교 전자컴퓨터 공학부 학사 졸업
2010년 전남대학교 전자컴퓨터 공학과 석사 졸업
<주관심분야 : 컴퓨터, 반도체, 디바이스 드라이버>

박 성 모(정회원)
1977년 서울대학교 전자과 학사 졸업
1979년 KAIST 전기및전자과 석사 졸업
1988년 North Carolina State Univ ECE 컴퓨터공학 박사 졸업
1979년~1984년 한국전자기술연구소 설계 개발부 선임연구원
1988년~1992년 Old Dominion Univ. ECE 조교수
1992년~현재 전남대학교 전자컴퓨터공학 교수
<주관심분야 : 컴퓨터, 반도체, VLSI, SoC>