NAND 플래시 파일 시스템 시뮬레이터 구현[1)]

옥동석[O], 이승환, 이태훈, 정기동
부산대학교 컴퓨터공학과
{ok3, cat, withsoul, kdchung}@pusan.ac.kr

Implementation of NAND Flash File System Simulator

Dong-Seok Ok[O], Seung-Hwan Lee, Tae-Hoon Lee, Ki-dong Chung
Dept. Computer Engineering, Pusan National University

요 약

최근 플래시 메모리는 임베디드 시스템과 휴대용 기기 등에서 많이 사용되고 있다. 그 중 NAND 플래시 메모리는 대용량화가 가능해 NOR 플래시 메모리보다 더 많이 사용되고 있다. NAND 플래시 파일 시스템을 개발하기에 앞서 파일 시스템의 할당과 가비지 컬렉션 정책을 시험하기 위해서, 또는 실제 플래시 메모리에서 수행하기 힘든 지움 횟수 측정 실험을 하기 위해서 플래시 파일 시스템 시뮬레이터를 구현하여 실험을 한다. 하지만 이 시뮬레이터는 제한된 성능 비교를 할 수 밖에 없는 약점을 지니고 있다. 이 때문에 어느 한 성능 개선을 위해 제안한 기법으로 인해서 다른 성능이 저하될 수 있지만 이를 간과할 수도 있다. 본 논문에서는 NAND 플래시 파일 시스템의 전체적인 성능 평가를 수행할 수 있는 시뮬레이터를 설계하고 구현한다.

1. 서 론

플래시 메모리는 비휘발성 저장장치로 최근 임베디드 시스템에서 각광받고 있다. 플래시 메모리는 NAND형과 NOR형으로 나눌 수 있다. NAND 플래시 메모리는 집접도가 낮은 NOR형에 비해 NAND는 고집적화가 가능하므로 제한된 크기 내에서 대용량화가 가능하다 NOR형은 RAM과 유사한 방식으로 접근 가능한 구조를 가지고 있기 때문에 코드 저장용으로 많이 사용되고, NAND는 고집적화가 가능해 대용량 플래시 메모리를 만들 수 있다는 점과 페이지 단위로 접근할 수 있기 때문에 많은 데이터를 한 번에 쓸 때 속도 면에서 유리하다는 점에서 데이터 저장 용도로 많이 사용한다

최근 플래시 파일 시스템에 관련된 이슈들로는 세 가지가 있다. 첫 번째로 빠른 마운트가 있다. 보통의 디스크 파일 시스템은 메타데이터를 디스크의 지정된 영역에 저장한다이 파일 시스템은 마운트 시 지정된 곳에서 메타데이터만 읽어서 파일 시스템의 데이터 구조를RAM 상에 구축한다. 이 경우 마운트 속도가 빠르다. 하지만 대표적인 플래시 파일 시스템인YAFFS나 JFFS2는 로그-구조 파일 시스템이므로 메타데이터도 플래시 메모리에 순차적으로 쓴다. 이 플래시 파일 시스템을 마운트하면 플래시 메모리 전체를 읽어 메타데이터를 찾아 파일 시스템 구조를 RAM상에 구축한다[1]. 이 구조 때문에 플래시 파일 시스템은 디스크 파일 시스템보다 상대적으로 느리다

두 번째로 지움 횟수 평준화가 있다 플래시 메모리는 구조상 블록 지움 횟수가 10,000에서 1,000,000 회로 제한된다 [2]. 이 횟수를 넘어간 블록은 더 이상 사용할 수 없고 이 플래시 메모리의 신뢰성이 저하된다 이러한 문제 때문에 플래시 파일 시스템은 별도로 지움 횟수 평준화를 위한 정책을 가지고 있고, 이를 위한 연구도 활발하게 진행되고 있다 하지만 지움 횟수 평준화를 위해서 많은 데이터 구조가 사용되고 연산량이 증가하기 때문에 램 사용률이 증가되고 쓰기 성능이 저하될 수 있다.

세 번째로 가비지 컬렉션이 있다 NAND 플래시 메모리의 지움 단위는 블록이다 지워야 할 페이지가 생기면 이 페이지를 바로 지우는 것이 아니라INVALID 표시를 해 두었다 나중에 한 번에 블록을 지우게 된다 이러한 특징 때문에 가비지 컬렉션을 하지 않으면 플래시 메모리의 사용 가능한 공간이 급속하게 줄어들게 된다. 가비지 컬렉션 정책은 크게 두 가지로 나눌 수 있는데 하나는 최대한 많은 페이지들을 회수하는 정책이고 다른 하나는 플래시 메모리 지움 횟수 정도를 균일화 하는 것이다.

본 논문에서 제안하는 시뮬레이터는 두 파일 시스템의 마운트 속도, 가비지 컬렉션 성능, 블록의 지움 횟수, 쓰기 연산의 성능을 비교할 수 있다 두 파일 시스템의 전체 성능을 평가함으로써 제안하고자 하는 기법의 성능과 약점을 동시에 파악할 수 있다.

2. 관련연구

2.1. 기존의 플래시 파일 시스템 시뮬레이터

플래시 메모리 지움 횟수나 특정 시점에서 플래시 메모리의 INVALID 페이지의 비율을 구하기란 매우 어렵다 이러한 현실

1) “이 논문은 2단계 두뇌한국21 사업에 의해 지원되었음”

적으로 수행하기 어려운 실험의 경우 시뮬레이터로 대체해서 실험을 할 수 있다[3][4]. 보통 지움 횟수 평준화 실험은 실제로 플래시 메모리에 일정 횟수 이상 지워야 하기 때문에 이를 실험하는 것은 시간이 많이 소요될 뿐 아니라 플래시 메모리의 수명을 소모해야하므로 실험 시 제약사항이 발생한다이 경우 시뮬레이터를 통해서 가상적으로NAND 플래시 메모리와 파일 시스템을 구현하여 실험을 수행할 수 있다 하지만 기존 플래시 파일 시스템 시뮬레이터의 경우 특정 성능만을 측정하기 위해 만들어지는 것이 대부분이다. 이 경우 자신이 제안한 특정 성능의 개선 기법이 다른 성능에 악영향을 미칠 수 있으나 알아차리지 못할 수 있다.

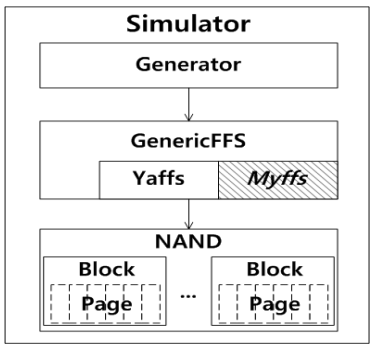
2.2. 요구사항

우리가 만들고자 하는 시뮬레이터에 요구되는 사항은 다음과 같다. 첫째, 우리가 실험하고자 하는 기법의 추가가 쉬워야 한다. 실험을 위해서 모든 파일 시스템을 구현할 필요는 없다 만약 기본적으로 제공하는YAFFS를 기반으로 해, 추가적인 기능을 구현하고자 한다면 추가적인 기능만을 구현함으로써 실험이 가능해야 한다.

둘째, 해당 기법의 성능 뿐 아니라 파일 시스템의 전반적인 성능평가도 가능해야 한다 우리가 이 시뮬레이터에 요구하는 성능 평가는 마운트 속도 쓰기 성능, INVALID 페이지 비율, 지움 횟수 측정 등이다.

셋째, 다양한 실험 방법을 제공해야 한다 임의의 크기의 파일 쓰기와 미리 준비된 연산의 시퀀스도 제공되어야 한다 플래시 파일 시스템의 성능을 평가하는 벤치마크 프로그램의 경우 여러 가지 트레이스를 제공하고 있다 그와 유사하게 여러 가지 워크로드를 가정해 트레이스가 제공되어야 한다

3. 시뮬레이터의 설계



(그림 1) 시뮬레이터의 구조

본 논문에서 제안하는 시뮬레이터의 전체적인 구조는(그림 1)과 같다. 이 시뮬레이터는 크게 시퀀스 발생기 플래시 파일 시스템, NAND 플래시 메모리로 구성되어 있다 실험 시 시퀀스 발생기인 Generator를 통해서 원하는 연산의 시퀀스를 만든다. 이 연산들은 파일 시스템에서 처리된다 파일 시스템은 Yaffs와 Myffs 2가지 구체화된 파일 시스템이 있고 시퀀스 발생기에서 발생한 연산은 두 파일 시스템 모두 동일하게 처리된다. 파일 시스템은 플래시 메모리를 가상화시킨NAND에 직접 연산을 수행한다.

3.1. 시뮬레이터의 시퀀스 발생기

시퀀스 발생기는 실험을 수행하는 사용자에게 랜덤 연산과 주어진 순서인 트레이스에 따라 연산을 수행하는 두 가지 실험 방법을 제공한다. 랜덤 방식은 쓰기 연산만을 수행하도록 하고 트레이스 방식은 쓰기, 읽기, 지움 연산을 모두 하도록 한다 시퀀스 발생기가 만들어낸 각 연산들은 플래시 파일 시스템으로 요청되고, 그 연산에 소요되는 시간을 반환받는다 이 반환받은 값을 파일에 기록하여 시뮬레이터가 각 연산에 걸린 시간을 각 연산에 소요되는 시간을 분석할 수 있도록 한다 하나의 연산이 끝나고 나면 시퀀스 발생기는NAND 플래시 메모리에 INVALID의 비율을 읽어 파일에 기록한다 이를 통해 시뮬레이터가 각 연산 중 가비지 컬렉션이 얼마나 효율적으로 작동했는지를 파악할 수 있다. 모든 연산이 끝나면 시퀀스 발생기는 두 파일 시스템을 마운트하여 마운트에 걸리는 시간을 측정화면에 출력한다.

3.2. 시뮬레이터의 플래시 파일 시스템

시뮬레이터는 비교 측정을 위해 두 가지 플래시 파일 시스템을 가진다. 하나는 시뮬레이터가 기본적으로 제공하는Yaffs이고 하나는 사용자가 구현해야하는 플래시 파일 시스템인 Myffs이다. 이 플래시 파일 시스템들은 공통의 인터페이스를 제공하기 위해 GenericFFS라는 추상 플래시 파일 시스템을 정의한다. 이 추상 플래시 파일 시스템은 시뮬레이터에 기본적인 파일 쓰기, 파일 읽기, 파일 지움, 마운트와 언마운트 연산을 제공한다. Yaffs와 Myffs는 이 인터페이스를 구현한다.

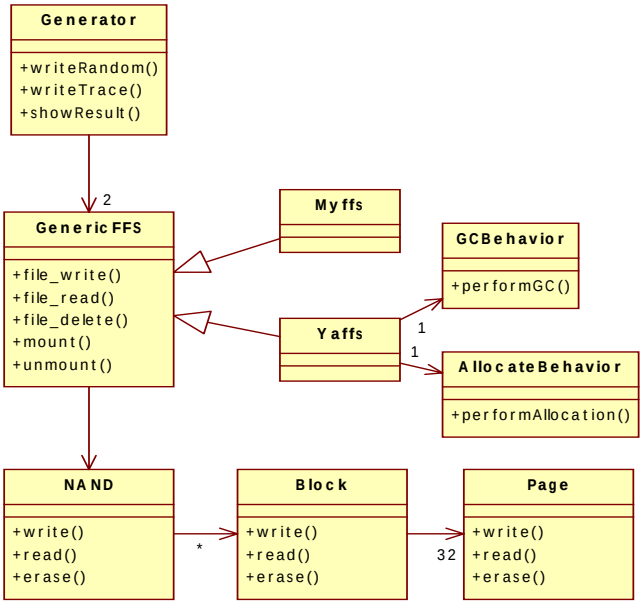
파일 시스템에서 할당과 가비지 컬렉션 정책은 분리하여 구현하도록 하였다. Yaffs의 경우 순차적인 할당 정책과 Greedy한 가비지 컬렉션 정책을 사용한다 하지만 이 Yaffs에서도 할당과 가비지 컬렉션 정책을 변형할 수 있어야 한다 따라서 할당은 AllocateBehavior, 가비지 컬렉션은 GCBehavior로 나누어서 인터페이스를 설계하였다 사용자가 이 AllocateBehavior와 GCBehavior를 구현함으로써 Yaffs의 할당과 가비지 컬렉션 정책을 변경할 수 있다.

3.3. 시뮬레이터의 NAND 플래시 메모리

플래시 파일 시스템의 쓰기, 읽기, 지움 연산은 플래시 메모리 상에서 이루어진다. 이를 위해서 가상의 NAND 플래시 메모리인 NAND와 Block, Page를 정의한다. 이 NAND 플래시 메모리 클래스들은 각각 쓰기, 읽기, 지움 연산을 정의한다 이 연산의 반환 값은 각 연산에 소요되는 시간이다 이 시간을 이용하여 시뮬레이터가 연산에 걸리는 시간을 측정할 수 있다 쓰기, 읽기, 지움 연산은 실제 NAND 플래시 메모리와 동일하게 쓰기, 읽기는 페이지 단위, 지움은 블록 단위이다.

위의 사항을 반영하여 설계한 NAND 플래시 파일 시스템 시뮬레이터의 클래스 다이어그램은(그림 2)와 같다. 이 클래

스 다이어그램은 각 클래스가 제공하는 인터페이스에 관해서만 기술되어 있다. 시뮬레이터의 세부적인 함수와 변수에 대해서는 따로 언급하지 않는다.



(그림 2) NAND 플래시 파일 시스템 시뮬레이터 설계 클래스 다이어그램

4. NAND 플래시 파일 시스템 시뮬레이터 구현

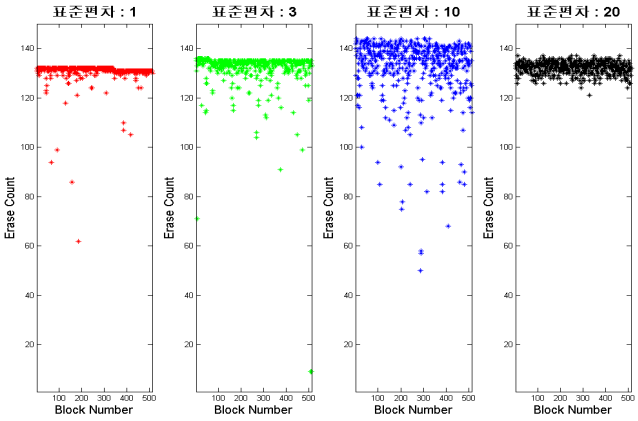
4.1. 시퀀스 발생기

NAND 플래시 파일 시스템 시뮬레이터는 설계에 따라 다형성을 구현할 수 있는 C++로 구현하였다. 먼저 시퀀스 발생기의 writeRandom은 연산의 대상인 Object ID, 파일의 사이즈를 랜덤하게 결정한다. 정규분포를 따르는 난수의 표준편차를 변화시켜 실험한 결과 표준편차가 10일 때 Yaffs의 지움 횟수 평준화가 잘 지켜지지 않았기 때문에 시퀀스 발생기가 사용하는 표준정규분포를 따르는 난수의 표준편차를 10으로 정했다. 각 표준편차에 따른 지움 횟수의 분포는 (그림 3)과 같다. writeTrace는 주어진 연산의 순서에 따라 연산을 수행한다. 시퀀스 발생기는 5가지의 기본적인 시퀀스를 가진다. 이 시퀀스들은 파일로 이루어져있고 writeTrace는 주어진 인자가 가리키는 파일을 읽어 이 파일에 기록된 연산의 종류 Object ID, 파일 사이즈를 바탕으로 연산을 요청한다. 해당 연산은 쓰기, 읽기, 지움 연산 세 가지이고 Object 개수는 100개, 평균 파일 사이즈는 10240Byte이다.

4.2. 파일시스템

시뮬레이터에서 실험 대상이 되는 모든 플래시 파일 시스템은 GenericFFS라는 추상 클래스를 상속받는다. 이 추상 클래스는 file_write(), file_read(), file_delete(), mount(), umount() 5가지의 순수 가상 함수를 정의하고 있다. 따라서 구현되는 파일 시스템들도 이 인터페이스에 따라 구현되어야 한다. 시뮬레이터가 기본적으로 가지는 플래시 파일 시스템은 Yaffs이다. 이 Yaffs는 Yaffs1의 소스 코드를 바탕으로 작성되었다[6]. 추가적으로 여러 가지 페이지 할당과 가비지 컬렉션 정책을 지원하기 위해서 각각에 대해 GCBehavior와 AllocateBehavior를 정의하였다. 기본적인 할당과 가비지 컬렉션은 Yaffs와 동일하지만 사용자가 새롭게 구현하여 사용할 수도 있다. Yaffs를 위해서 GreedyGC와 DefaultAllocator 두 가지의 기본적인 가비지 컬렉션 정책과 페이지 할당 정책을 구현하였다.

사용자가 구현할 Myffs는 추상 클래스인 GenericFFS의 인터페이스에 따라 구현하고 그 외의 함수와 변수는 사용자의 목적에 맞게 구현한다. 기본적으로 제공하는 Yaffs를 상속하여 Yaffs와 비슷한 파일 시스템을 쉽게 구현할 수도 있다. 이 경우 목적에 맞게 일부 함수나 GCBehavior, AllocateBehavior만 수정하여 사용할 수 있다.



(그림 3) 정규분포의 표준편차에 따른 난수 사용 시 각 블록의 지움 횟수 분포

4.3. NAND 플래시 메모리

NAND는 NAND 플래시 Small Block을 기준으로 작성되었다. 세부적인 사항은 삼성 NAND 플래시 메모리 K9F1208U0B의 기술문서를 참고하였다[7]. 이 문서에 따라 플래시 메모리의 쓰기 속도는 200μs, 읽기 속도는 15μs, 블록 삭제 속도는 2ms로 정의하였다. 각 쓰기, 읽기, 지움 연산은 해당 연산에 소요되는 시간을 반환한다. 연산을 발생시키는 시퀀스 발생기는 이 시간을 반환받아 연산에 걸리는 시간을 측정기록하여 성능을 평가할 수 있다.

파일 시스템의 파일 쓰기, 읽기, 지움 연산은 NAND가 제공하는 인터페이스를 사용하여 이루어진다. NAND는 write(), read(), erase() 연산을 제공한다. NAND 플래시 메모리의 쓰기와 읽기는 페이지 단위로 이루어지고 지움 연산은 블록단위로 이루어진다. 따라서 NAND의 write()와 read()의 인자는 페이지 번호이고 erase()의 인자는 블록 번호이다.

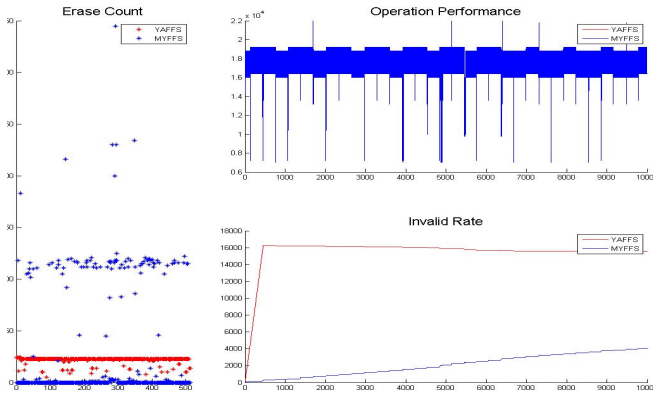
NAND 플래시 메모리의 기본적인 크기는16MB이다. 이 크기는 실험 목적에 따라 변경할 수 있다. NAND를 생생할 때 인자는 NAND에 포함되는 블록의 개수이다. 이 개수를 지정함으로써 NAND의 크기를 정할 수 있다. NAND에 포함되는

Block 역시 NAND와 마찬가지로 write(), read(), erase() 연산을 제공한다. Block은 32개의 Page로 이루어져있다. write()와 read() 연산의 인자는 블록 내의 페이지 번호이다. 이 페이지 번호가 가리키는 페이지에 접근해서 페이지의 연산을 수행한다. Block의 erase() 연산은 해당 Block의 모든 Page를 삭제한다. Block의 erase()가 호출될 때마다 Block의 eraseCount 변수가 1씩 증가한다. 이 eraseCount 변수를 이용하여 각 블록의 지움 횟수를 파악할 수 있다. 지움 연산이 호출되면 설계에 따라 블록 지움에 걸리는 시간을 반환해야하므로 지움 연산에 걸리는 시간을 가지고 있다. Page도 NAND, Block과 마찬가지로 write, read, erase 연산을 제공한다. 실험 목적을 가지는 시뮬레이터이므로 실제 NAND 플래시 메모리처럼 데이터 부분을 가지지는 않는다. 다만 파일 시스템이 Spare 영역에 Object ID나 Invalid 표시를 하므로 Spare 영역 16Byte를 가진다. 그 외에 쓰기, 읽기 연산 수행의 결과로 연산에 걸리는 속도를 반환하기 위해서 쓰기, 읽기에 걸리는 시간을 가지고 있다.

4.4. 파일 시스템 평가 도구

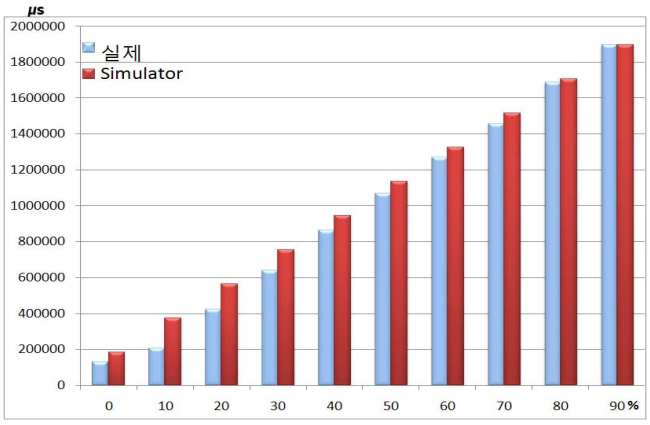
위와 같은 과정을 거쳐서 비교하는 두 파일 시스템의 블록당 지움 횟수, 연산 당 걸리는 시간, 연산 수행 시 플래시 메모리의 INVALID 페이지의 비율들이 파일에 출력된다. 마운트 시간은 각 파일 시스템 당 하나의 시간 값을 가지므로 화면에 출력한다.

이 출력된 파일을 읽어 MATLAB을 이용해 도식화한다. 이 그림은 (그림 4)와 같다. 이 세 가지 그래프와 콘솔 창에 출력되는 마운트 시간으로 파일 시스템 간 성능을 비교할 수 있다.



(그림 4) 시뮬레이터의 파일 시스템 성능평가 결과

시뮬레이터의 가상 플래시 메모리와 실제 플래시 메모리의 속도 차이를 알아보기 위해 시뮬레이터와 실제 플래시 메모리에서의 플래시 메모리 이용률에 따른YAFFS의 마운트 시간을 비교해보았다. 그 결과 (그림 5)와 플래시 메모리 이용률이 10%일 때 166ms로 가장 차이가 많이 났고, 90%일 때는 0.3ms로 가장 작은 차이가 발생하였다. 우리는 실험 시 많은 양의 데이터를 플래시 메모리에 쓰므로 플래시 메모리 이용률이 클 것이다. 따라서 시뮬레이터와 실제 플래시 메모리 간의 차이는 거의 없다고 봐도 될 것이다.



(그림 5) 실제 NAND 플래시 메모리와 시뮬레이터에서 플래시 메모리 이용률에 따른 YAFFS의 마운트 성능 비교

5. 결론

본 논문에서는 기존 NAND 플래시 파일 시스템 시뮬레이터의 제한된 기능을 개선하여 플래시 메모리의 지움 횟수쓰기/읽기/지움 연산의 성능, 마운트 시간, 가비지 컬렉션 성능을 모두 비교할 수 있는 시뮬레이터를 설계 구현하였다. 이 시뮬레이터로 실제 파일 시스템의 성능을 측정할 수 없지만 파일 시스템 설계 단계에서 할당과 가비지 컬렉션 정책 수립 시 도움을 줄 수 있다.

참고문헌

[1] 진종원, 이태훈, 정기동, “빠른 마운트와 복구를 지원하는 NAND 플래시 파일 시스템 설계”, *한국정보과학회 2007가을학술발표논문집*, 제 34권, 제 2호(B), pp.404~ 407, 10월. 2007년

[2] Eran Gal and Sivan Toledo, “Algorithms and data structures for flash memories”, *ACM Computing Surveys*, 37(2), 2005.

[3] 이태훈, 이상기, 정기동, "임베디드 플래시 파일 시스템을 위한 플레인 지움 정책", *한국정보처리학회 제32회 추계학술발표회논문집*, pp 778-78, 2005년

[4] 전승진, 공기석, 황달연, “플래시 메모리 파일 시스템의 지움 정책 개선에 관한 연구”, *한국정보과학회 2003가을학술발표논문집*, 제 30권, 제 2호, pp. 289~291, 10월 2003년

[5] 박상오, 김성조, "리눅스 기반의 NAND 플래시 메모리 파일 시스템에 대한 성능 측정 도구 설계", *한국정보과학회 2005가을학술발표논문집*, 제 32권, 제 2호, pp.844~846, 11월, 2005년

[6] YAFFS, http://www.yaffs.net/

[7] Samsung "64Mx8Bit NAND Flash Memory K9F1208U0B", 2005