NAND 플래시 메모리 파일 시스템에서 빠른 연산을 위한 설계

(Design of Fast Operation Method in NAND Flash Memory File System)

진 종 원 † 이 태 훈 †
(Jongwon Jin) (Taehoon Lee)

정 기 동 ††
(Kidong Chung)

요 약 플래시 메모리는 비휘발성, 저전력, 빠른 입출력, 충격에 강함 등과 같은 많은 장점을 가지고 있으며 모바일 기기에서의 저장 매체로 사용이 증가 되고 있다. 하지만 제자리 덮어쓰기가 불가능하고 지움 연산의 단위가 크다는 제약 및 블록의 지움 횟수 제한이 있다. 이러한 제약을 극복하기 위해 YAFFS와 같은 로그 구조 기반의 플래시 파일 시스템들이 개발 되었다. 그러나 쓰기 연산을 위한 공간 요청이 발생할 때나 지움 대상 블록을 선정할 때 순차적으로 블록 정보를 검색하여 할당 및 지움 연산을 수행한다. 이러한 순차적인 블록 접근 방식은 플래시 메모리의 사용량이 증가함에 따라 접근 시간이 증가될 수 있다. 그리고 블록 지움 연산을 수행하는 시기를 결정하여 불필요한 지움 연산 대상 블록을 찾는 시간을 최소화하고 충분한 플래시 메모리의 빈 공간을 유지 하여야 한다.

본 논문에서는 이러한 문제점을 해결하기 위해 로그 구조 기반의 NAND 플래시 메모리 파일시스템의 빠른 연산을 위한 기법들을 제안한다. 제안된 기법은 YAFFS 상에서 구현되었으며, 제안한 기법들을 실험을 통해 비교·분석하였다. 제안된 기법은 기존의 성능과 비교해 빠른 연산 성능 향상을 보였다.

키워드 : 플래시메모리, 파일시스템, 낸드플래시, 할당자

Abstract Flash memory is widely used in embedded systems because of its benefits such as non-volatile, shock resistant, and low power consumption. But NAND flash memory suffers from out-place-update, limited erase cycles, and page based read/write operations. To solve these problems, log-structured filesystem was proposed such as YAFFS. However, YAFFS sequentially retrieves an array of all block information to allocate free block for a write operation. Also before the write operation, YAFFS read the array of block information to find invalid block for erase. These could reduce the performance of the filesystem.

This paper suggests fast operation method for NAND flash filesystem that solves the above-mentioned problems. We implemented the proposed methods in YAFFS. And we measured the performance compared with the original technique.

Key words : Flash memory, Filesystem, NANDflash, Allocator

1. 서 론

PDA, PMP와 RFID 리더 등의 모바일 기기에서의 저장매체로 플래시 메모리 사용이 증가하고 있다. 플래시 메모리는 EEPROM의 일종으로 비휘발성, 저전력, 충격에 강함 및 낮은 가격 등의 장점이 있다. 또한 하드디스크가 데이터의 위치에 따른 헤드의 탐색 거리에 따라 성능에 큰 영향을 주지만 플래시 메모리의 경우 위치에 상관없는 빠른 입출력 속도를 가지고 있다[1].

본 논문에서 대상으로 하고 있는 저장 매체인 플래시 메모리는 블록으로 구성이 되며 각 블록은 다시 페이지로 구성이 된다. 그리고 페이지는 data영역과 spare영역으로 구분되어 구성이 되고 읽기 속도와 쓰기 속도는 표 1과 같다.

표 1에서 보는 것과 같이 플래시 메모리는 읽기와 쓰기 처리시간이 각각 다르고 지움 시간이 크다. 그리고 플래시 메모리는 제자리 덮어쓰기가 불가능하고 지움 연산의 단위가 읽기·쓰기 단위보다 크다는 단점 및 블록의 지움 횟수 제한이 있다[2].

표 1 저장 매체별 처리속도

저장매체	처리시간(512Bytes)		
	Read	Write	Erase
DRAM	2.56us	2.56us	N/A
NOR flash	14.4us	3.53ms	1.2s(128KB)
NAND flash	35.9us	226us	2ms(16KB)
DISK	12.4ms	12.4ms	N/A

· 이 논문은 2단계 두뇌한국21사업에 의하여 지원되었음
· This work was supported by the Brain Korea 21 Project in 2007.
· 이 논문은 2007 한국컴퓨터종합학술대회에서 'NAND 플래시 메모리 파일 시스템의 빠른 연산을 위한 설계 및 성능 평가'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 부산대학교 컴퓨터공학과
waller@pusan.ac.kr
withsoul@pusan.ac.kr
†† 종신회원 : 부산대학교 컴퓨터공학과 교수
kdchung@pusan.ac.kr

논문접수 : 2007년 10월 2일
심사완료 : 2007년 12월 28일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제14권 제1호(2008.2)

이러한 플래시 메모리의 단점을 극복하기 위해 YAFFS[3]와 같은 플래시 메모리 파일 시스템이 개발되었다. YAFFS는 로그 구조 기반의 파일 시스템으로 NAND 플래시 메모리 사용을 위해 개발되었다. YAFFS는 공간 할당을 위해서는 순차적인 방법으로 블록 상태를 검색하여 빈 블록을 할당한다. 순차적인 블록 상태를 검색하여 할당하는 방법은 플래시 메모리 사용량이 높아서 빈 블록이 연속적이지 않을 경우 빈 블록 할당에 많은 시간이 소모 된다. 또한 블록 지움 횟수에 제한이 있기 때문에 균등화를 고려하여 블록 할당을 수행하여야 한다. 그리고 블록 지움 연산 수행 시 지움 대상 블록을 순차적으로 검사하여 많은 시간을 소모하게 된다. 또한 YAFFS는 블록 지움 연산을 수행할 때 페이지 단위로 지움 대상 블록을 검사하여 시간을 많이 소모한다. 그리고 플래시 메모리의 사용공간을 고려하지 않고 수행하여 성능이 저하된다.

이러한 문제점을 해결하기 위해 본 논문에서는 로그 구조 기반의 NAND 플래시 메모리 파일시스템의 빠른 연산을 위한 관리 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 로그 구조 기반의 NAND 플래시 메모리 파일 시스템인 YAFFS에 대해 기술하고, 3장에서는 효율적인 관리 기법들을 제안한다. 4장에서는 제안한 기법들의 성능을 실험을 통해 비교·분석하고 5장에서 결론을 맺는다.

2. YAFFS

NAND 플래시 메모리를 위한 대표적인 파일 시스템으로는 YAFFS(Yet Another Flash File System)가 있다. YAFFS는 2002년 Alepha one 사에서 개발된 NAND 플래시 전용 파일 시스템이다. YAFFS는 로그 구조를 기반으로 하여 플래시 메모리에 대한 갱신 연산을 추가 연산으로 변형하여 처리하는 외부 갱신 기법을 사용한다. 이를 통해 플래시 메모리의 제자리 덮어쓰기가 되지 않는 문제점을 해결하였다[4].

그러나 쓰기 연산을 위한 공간 요청 발생 시, 빈 블록을 찾기 위해서 블록 상태 정보를 순차적으로 검색하게 된다. 또한 순차적인 검색을 통한 할당으로 인해 블록 사용 횟수에 대한 고려 없이 할당 하게 되므로 플래시 메모리의 균등사용을 지원하지 못한다.

그림 1은 YAFFS가 사용하는 자료 구조를 보여주고 있다. RAM상의 allocationBlock과 allocationPage는 현재 사용 중인 블록과 블록 내의 페이지 번호를 나타낸다. Block_Info 구조체는 마운팅 과정에서 플래시 메모리를 읽어 메모리(RAM) 상에 구축된다. 이러한 구조로 인해 플래시 메모리의 사용량이 높아 빈 블록이 플래시 메모리 상에 많이 흩어져 있는 경우, 빈 공간 할당 시

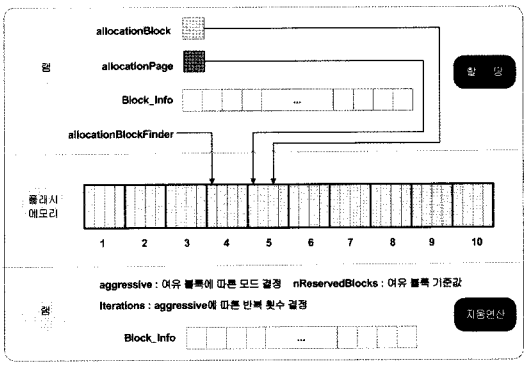

그림 1 YAFFS 자료구조

빈 블록 검색에 많은 시간이 소요된다. 또한 지움 횟수에 대한 고려 없이 순차적인 할당을 수행하여 플래시 메모리를 균등하게 사용할 수 없다.

그리고 블록 지움 연산 수행시 지움 대상 블록을 순차적으로 검사하는 작업을 수행하여 많은 시간을 소모하게 된다. 또한 YAFFS 파일시스템은 블록 지움 연산을 수행할 때 페이지 단위로 지움 대상 블록을 검사하여 시간을 많이 소모한다. 그리고 지움 대상 블록을 선택할 때 부분적인 영역을 검사하는 방법을 사용하므로 전체 블록에서 적합한 지움 대상 블록을 찾지 못한다.

본 논문에서는 이러한 문제점을 해결하기 위해 로그 구조 기반의 NAND 플래시 파일 시스템의 빠른 연산을 위한 관리 기법을 제안한다.

3. 제안하는 관리 기법

3.1 균등 사용을 고려한 빠른 할당 기법

YAFFS 파일시스템의 경우 빈 블록 할당을 위해서 각 블록의 상태 정보를 저장하고 있는 Block_Info라는 구조체에서 블록을 순차적으로 읽어 온다. 이렇게 순차적인 구조는 블록의 빈 공간이 부족하여 빈 블록 간의 거리가 멀어지게 되면 빈 블록을 찾기 위해 불필요한 자원을 사용하게 된다. 이러한 순차적인 검색의 문제를 해결하기 위해 빈 블록 정보를 연결 리스트를 사용하여 유지한다.

그림 2는 공간 할당을 위한 빈 블록 연결 리스트를 보여주고 있다. 연결 리스트의 각 노드는 다음 빈 블록의 위치인 'blocknumber'와 다음 노드를 가리키는 포인터인 'next' 그리고 연속된 블록을 나타내기 위한 'continuity'로 구성된다. 그리고 빈 블록 연결 리스트는 블록 지움 연산을 수행한 후에 블록 지움 연산 수행 횟수를 기록하여 블록 지움 연산 수행 횟수별로 연결 리스트를 유지한다. 이 정보들을 이용하여 블록 할당 요청의 경우 블록 지움 횟수가 작은 연결 리스트의 빈 블록을

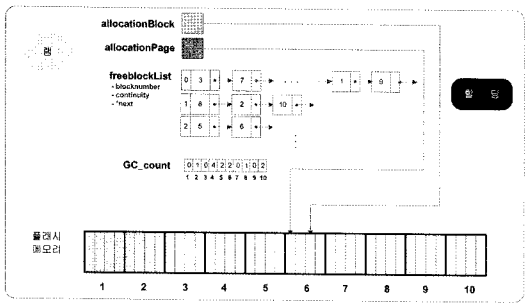

그림 2 균등 사용 및 빠른 연산을 위한 구조

먼저 할당하게 함으로서 균등한 블록 사용을 보장해 줄 수 있게 된다.

위와 같은 방법으로 제안한 정보를 유지할 경우 연결 리스트를 위한 추가적인 메모리(RAM)가 필요하게 되지만 플래시 메모리 용량이 커지고 사용량이 많이 질수록 선형적인 구조에 비해 연결 리스트로의 정보 유지가 효율적이게 된다. 또한 블록 지움 횟수별에 따라 우선순위별 리스트 구조를 구성함으로서 삽입/삭제 시에 효율적으로 관리할 수 있다. 또한 플래시 메모리는 기존 하드디스크 파일시스템과 달리 위치에 상관없는 빠른 입출력을 할 수 있다는 점을 고려할 때 연결 리스트를 이용한 구조로 인해 하드디스크 파일시스템에서 고려해야 했던 파일 위치에 따른 헤더 이동에 관한 요소는 고려하지 않아도 되므로 효율적이다.

3.2 플래시 메모리 블록 지움 연산의 개선

YAFFS 파일시스템에서 블록 지움 연산은 쓰기 작업이 들어오게 되면 페이지 단위로 지움 대상 블록을 검사한다. 즉, 해당 파일의 오브젝트헤더(Object Header) 정보가 변경이 일어나게 되는 데이터 수정, 크기변경 등의 작업을 수행하게 되면 새로운 데이터를 기록하거나 변경된 오브젝트헤더를 페이지 단위로 기록하면서 지움 대상 블록 검사를 수행한다. 이때 빈 블록이 확보되어야 할 최소의 값인 nReserved Blocks값 보다 작아 충분한 공간이 없을 시에는 전체 블록을 검사하여 가장 무효페이지가 많은 블록(Dirtiest block)을 찾아서 블록 지움 연산을 수행하게 되고, 충분한 공간이 있을 때에는 특정 iteration값을 지정하고 매 페이지 쓰기 연산을 수행할 때마다 iteration만큼의 블록을 검사하여 지움 대상 블록을 검사한다.

주기적으로 매번 확인하는 방식의 지움 대상 블록 검사는 지움 대상 블록 검사를 하는데 파일 시스템에 많은 부하를 생기게 하고 또한 충분한 공간이 없는 상황에서는 전체 블록을 검사하는 부하가 시스템에 걸리게 된다. 이러한 성능 저하를 가져오게 되는 문제를 해결하기 위하여 본 논문은 블록 지움 연산의 대상이 되는 블록의 무효페이지의 수를 유지하면서 무효페이지가 많은 블록을 우선 배치하여 연결 리스트 구조로 관리한다. 이러한 연결 리스트 구조로 불필요한 지움 대상 블록의 검사 없이 수행할 수 있게 된다.

그림 3은 블록 지움 연산을 효율적으로 수행하기 위한 지움 대상 블록의 연결 리스트 구조를 보여 주고 있다. 연결 리스트의 각 노드는 다음 지움 대상 블록의 위치를 나타내는 'blocknumber', 다음 노드를 가리키는 포인터인 'next'와 블록에서의 무효페이지 정보의 수를 나타내는 'invalid_count'로 구성된다. 이를 통하여 전체 플래시 메모리에서 가장 무효페이지가 가장 많은 블록을 지움 연산을 수행할 수 있게 한다.

또한 지움 연산을 수행하는 시기가 중요하다. YAFFS 파일시스템은 매번 파일객체 쓰기/갱신 작업 시에 페이지단위로 지움 대상 블록을 검사를 하고 파일 시스템에서 정의하고 있는 'YAFFS_PASSIVE_GC_CHUNKS+1' 값보다 유효페이지가 작으면 지움 연산을 수행한다. 매번 검사를 통한 방법은 검사 시간에 많은 비용을 소모하게 된다. 그러나 제안한 방법으로 무효페이지가 많은 블록을 우선순위로 관리하면 무효페이지가 많은 블록을 먼저 제거하여 불필요한 검색 시간을 크게 줄일 수 있다. 이 방법은 블록 지움 정책에서 Greedy하게 블록을 선택하는 방법으로 Cost-benefit 등의 기법에도 적용할 수 있다.

이렇게 구조를 변경 하게 되었을 때 기존의 YAFFS 파일시스템과 같이 매번 검사를 수행하는 것이 아니라 빈 블록 확보가 필요한 시점에 블록 지움 연산을 통해 블록을 할당한다. 본 논문에서는 새로운 블록을 할당 받을 때에 플래시 메모리의 사용량을 보고 사용량별 무효페이지 수의 임계값(threshold)을 가지는 테이블을 유지하여 플래시 메모리의 사용량이 작을 때에는 해당 블록이 무효페이지가 많은 블록에 대해서만 지움 연산을 수행하고 플래시 메모리 사용량이 많아지면 무효페이지에

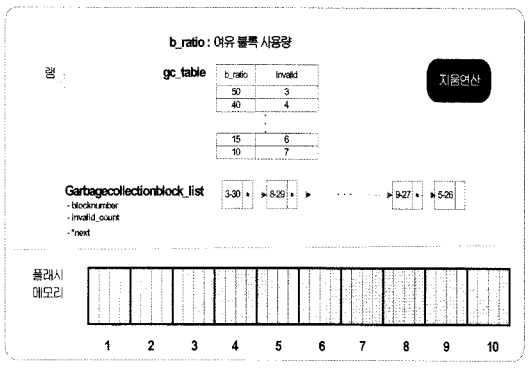

그림 3 블록 지움 연산 향상을 위한 구조

대한 임계값을 감소하게 하여 빈 블록 확보를 더 많이 할 수 있도록 하였다. 앞서 설명한 플래시 메모리 사용량별 무효페이지가 많은 블록을 선택하는 무효페이지 임계값은 임의로 지정하였다.

3.3 성능 향상

제안하는 관리기법은 메모리(RAM) 상에 필요한 데이터를 미리 구축해놓고 관리함으로서 빠른 성능을 제공하고 있다. 그러나 NAND 플래시 메모리를 사용하는 시스템의 대부분은 소량의 메모리로 동작을 하는 임베디드 시스템임이므로 메모리 사용량을 최소화해야 한다.

제안하는 관리기법에서 균등사용을 고려한 할당 기법의 경우, 연속적인 공간에 대한 메모리 사용의 경우 연속된 블록을 여러 개의 노드로 유지 하지 않고 하나의 노드로 유지할 수 있도록 연속성에 대한 값인 'continuity'를 기록하여 하나의 노드로 여러 개의 연속적인 빈 블록 정보를 유지할 수 있도록 하였다. 이로 인해 메모리 효율을 증가 시킬 수 있다.

또한 블록 지움 연산 개선 기법을 적용 하는 경우에 무효페이지의 정보에 따른 연결 리스트를 관리하는 과부하가 생기게 된다. 이를 최소화하기 위해서 플래시 메모리 사용량에 따른 무효페이지 테이블(gc_table)을 이용하여 테이블에 따른 임계값 이상의 블록만 관리 하여 관리 비용을 최소화 하였다. 여기서 지움 대상이 되는 블록을 연결 리스트로 유지하는 메모리 사용에 대한 비용은 지움 대상 블록에 대해서만 생성하고 지움 연산 수행 후 제거되어 메모리 사용에 부담을 최소화할 수 있다.

4. 성능 평가

제안하는 기법들을 실제 실험을 통해 성능을 비교하고 분석하였다.

실험을 위해 휴인스에서 개발된 PXA255-Ⅲ 임베디드 보드를 사용하였다. 삼성에서 개발된 64MB NAND 플래시 메모리를 사용하였으며 YAFFS에 제안한 기법들을 구현하여 실험하였다. 플래시 메모리는 소블록 NAND 플래시 메모리로 크기를 10MB 분할하여 사용하였고 운영체제로는 리눅스 2.4.19를 사용하였다.

표 2 PXA255-Pro III 임베디드 보드[5]

항 목		명 세
CPU		Intel PXA255(Xscale)
SDRAM		128 MB SDRAM
Flash Memory	NOR	32 MB (Intel)
	NAND	64 MB (Samsung)

실험보드의 사양은 표 2와 같다.

4.1 실험 방법

- 할당 성능 비교: 빈 블록이 연속적으로 분포하는 경우와 그렇지 않은 경우를 만들어 성능을 측정 하였다. 빈 블록의 경우 시스템의 사용에 따라서 일정하지 않게 분포하게 된다. 실험 시에는 빈 블록을 일정한 블록 단위로 분포하게 하여 성능을 측정하였다. 20개의 빈 블록과 20개의 사용 중인 블록들을 차례대로 분포하게 하여 Scan시에 수행하게 되는 플래시 메모리의 상태를 임의로 정하였다.
- 블록 지움 연산 성능 비교 : 일정크기(256Bytes~10 Kbytes)의 파일을 생성하고 삭제하는 트랙잭션을 플래시 메모리 사용량을 증가 시키면서 수행하였다. 파일명, 파일크기, 파일 생성 개수 등을 인자로 받아서 파일을 생성하는 프로그램을 이용해서 파일을 생성하고 파일을 삭제하는 성능을 측정하였다.
- 메모리 사용량 비교 : 블록 지움 연산 성능 비교와 같이 일정 크기 파일들을 생성, 삭제하면서 기존 YAFFS, 성능 향상을 적용하지 않은 방법과 성능 향상을 적용한 방법을 플래시 메모리 사용률별 메모리 사용량을 측정하였다.

4.2 실험 결과

앞서 설명한 성능 비교를 한 결과를 다음과 같이 그림으로 나타내었다.

그림 4는 플래시 메모리에 빈 블록을 일정 블록 단위로 설정을 하여 성능을 측정하였다. 본 논문에서는 20개 블록 단위로 빈 블록을 설정하였다. 그림 4를 보면 기존의 YAFFS의 경우 빈 블록이 연속적이지 않은 경우에 할당시간이 증가하는 것을 확인할 수 있다. 이러한 설정으로 기존의 방법에 비해 평균 10%의 성능 향상을 보였다. 플래시 메모리의 용량이 커지고 사용량이 많아질수록 빈 블록 할당을 위한 블록간 거리에 대한 검색 오버헤드가 커지게 되므로 더 많은 검색 시간이 필요하다. 그러므로 제안한 기법은 NAND 플래시 메모리의 크기에 비례하여 성능향상을 기대할 수 있다.

그림 5는 플래시 메모리 사용량에 따른 지움 대상 블록을 호출하는 누적횟수를 나타낸 것이다. 기존의 방법은 지움 연산을 수행하기 위해서 지움 연산의 대상 블록을 호출하여 검사한다. 이렇게 지움 대상 블록을 검사하는 작업은 실제 지움 연산을 수행하는 횟수에 비해 빈번하게 이루어진다. 그러나 제안한 방법의 경우 Block_info에서 정보를 읽어 오는 것이 아니라 무효페이지 수에 대한 정보를 통한 연결 리스트로 정보를 유지하고 있으므로 블록 할당을 하면서 블록 지움 대상 블록에 대한 확인을 하므로 블록 호출 횟수가 기존의 YAFFS에 비해 약 6%에 불과 하였다.

그림 6은 플래시 메모리의 사용률별 블록 지움 대상 블록을 선택하는 시간을 나타내고 있는데 기존의 YAFFS의

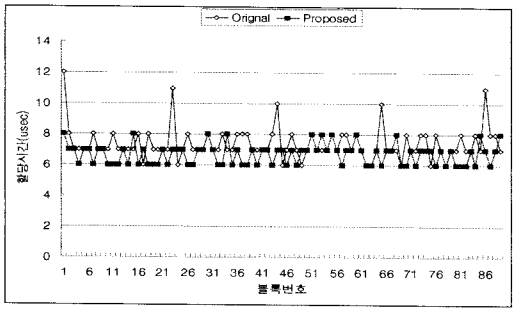

그림 4 할당 시간 비교

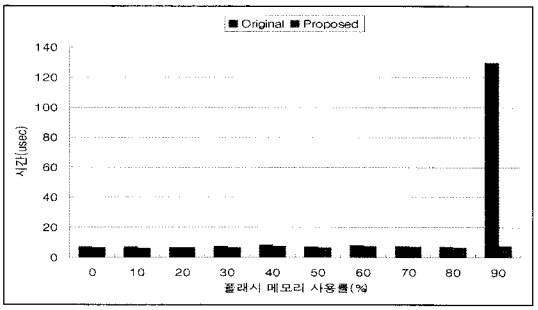

그림 6 사용률별 지움 연산 수행 시간 비교

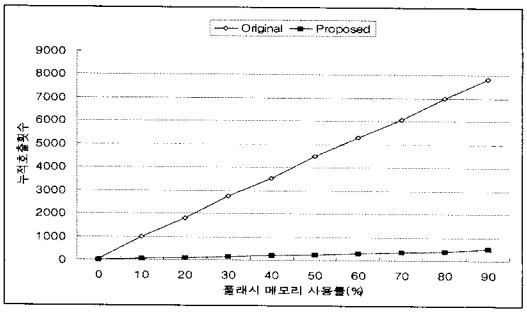

그림 5 지움 연산 누적호출횟수 비교

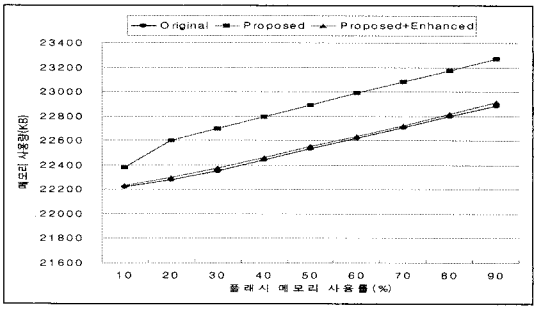

그림 7 메모리 사용량 비교

경우 사용률이 높아져 공간이 부족하게 되면 전체공간을 검사하는 상태로 되어 지움 대상 블록을 선택하는데 많은 시간이 걸리게 된다. 그러나 제안한 기법은 플래시 메모리 사용량이 증가하는 것과 관계없이 일정한 시간이 걸리는 것을 알 수 있다.

그림 7은 플래시 메모리에 사용되는 메모리(RAM)를 플래시 메모리 사용률별로 나타내고 있다. 실험에서 사용한 메모리는 128Mbytes 메모리로 부팅 시에 21.3~21.6Mbytes의 메모리를 사용한다. 파일 시스템의 플래시 메모리 사용량을 늘리면서 메모리 사용량을 확인하여 비교하였다. 실험은 YAFFS(original), 제안한 방법(proposed)과 성능향상이 적용된 제안한 방법(proposed +enhanced)을 비교하였다. 제안한 방법은 YAFFS에 비해 추가 메모리 사용이 필요로 하지만 메모리 사용량을 최소화할 수 있도록 하여 성능향상이 적용된 방법은 적용되지 않은 기법에 비하여 약 3.5~7.9%에 불과한 메모리 사용량을 볼 수 있었다. 단, 실험은 플래시 메모리 사용량이 0%부터 사용량을 증가시키면서 측정하여 마운팅시에 빈 블록 리스트를 만드는 비용이 최소화 된 것으로 충분히 사용된 플래시 메모리의 경우 마운팅시에 빈 블록 연결리스트를 만들기 위한 메모리 사용이 증가할 수 있다.

5. 결 론

본 논문은 로그 구조 기반의 NAND 플래시 파일시스템을 위한 효율적인 관리 기법들을 제안하였다. 빠른 할당을 위해 연결 리스트 구조를 이용하였고 이에 플래시 메모리의 블록 지움 횟수를 포함시켜 균등화된 사용이 가능하게 하였다. 그리고 플래시 메모리 블록 지움 연산에 대한 연결 리스트를 유지하고 블록 지움 연산이 필요한 블록을 검사하는 과정의 오버헤드를 제거하여 플래시 메모리의 사용량에 관계없이 빠르게 빈 블록 확보를 할 수 있게 하였다. 또한 제안한 구조를 유지하기 위해 필요한 메모리 사용량을 최소화 하였다.

참 고 문 헌

[1] F. Douglis, R. Caceres, F. Kaashoek, K. Li, B. Marsh, and J. A. Tauber, "Storage Alternatives for Mobile Computers,", pp.25-37, Proceedings of the 1st Symposium on Operating Systems Design and Implementation, 1994.

[2] 백승재, 최종무, "플래시 메모리 파일 시스템을 위한 순수도 기반 페이지 할당 기법에 대한 연구", 정보처리학회논문지 A 제13-A권 제5호, 2006.10.

[3] Yaffs Spec, http://www.aleph1.co.uk/taxonomy/term/31

[4] 박송화, 이태훈, 정기동, "pp. 151-153, 임베디드 기기를 위한 NAND 플래시 파일 시스템의 설계", 한국컴퓨터종합학술대회, 논문집(A) 제33권 1호, 2006.

[5] (주) 휴인스 기술연구소, "Intel PXA255와 임베디드 리눅스 응용 [제2판]", 홍릉과학출판사, 2004.