J Korea Industr Inf Syst Res Volume 18 Number 2
http://dx.doi.org/10.9723/jksiis.2013.18.2.027

NAND 플래시 파일시스템의 I/O 스케줄러 성능분석†

(A Performance Analysis of I/O Scheduler for NAND Flash File System)

이 영 석*, 이 창 희**, 정 경 호***, 김 용 환****, 안 광 선*****
(Yeongseok Lee, Changhee Lee, Kyungho Chung, Yonghwan Kim, and Kwangseon Ahn)

요 약 대용량의 NAND 플래시 메모리가 출시됨으로써, 다양한 용도로 사용이 가능해 졌다. 특히 모바일기기의 멀티미디어 기능 확장으로 인해 대용량 NAND 플래시 메모리의 수요가 증가하고 있다. YAFFS2, NILFS2, JFFS2 파일시스템은 NAND 플래시 메모리 전용 파일시스템이다. 본 논문에서는 각 3개의 파일시스템에 4개의 I/O scheduler : CFQ(Complete Fair Queuing) I/O scheduler, NOOP(No Operation) I/O scheduler, Anticipatory I/O scheduler, Deadline I/O scheduler에 대한 순차적인 읽기, 쓰기 성능을 분석하였다. JFFS2 파일시스템 상에서의 Anticipatory I/O scheduler가 다른 I/O scheduler보다 쓰기 8%, 읽기 1.5% 이상 시간이 단축되었다. YAFFS2 파일시스템상에서는 4개의 I/O scheduler 시간이 일정하다. NILFS2 파일시스템에서는 Deadline I/O scheduler가 다른 I/O scheduler보다 쓰기 2%, NOOP I/O scheduler가 읽기 6% 정도 시간이 단축 된다.

핵심주제어 : 낸드 플래시 메모리, I/O 스케줄러

Abstract NAND Flash Memory has been used in several devices by low cost and high capacity, and the demand for mass NAND Flash Memory has increased due to the multimedia extension of mobile devices. The JFFS2, NILFS2, and YAFFS2 file systems are used mainly in NAND Flash Memory. In this paper, the performance of Sequential read/write of the 3 file systems are analyzed for the 4 I/O schedulers : CFQ(Complete Fair Queuing) I/O scheduler, NOOP(No Operation) I/O scheduler, Anticipatory I/O scheduler, and Deadline I/O scheduler. In JFFS2 file system, Anticipatory I/O scheduler has the best performance by 8% decreasing speed in writing time and 1.5% decreasing speed in reading time compared to the other I/O scheduler. In YAFFS2 file system, it results are similar to performance in reading and writing for the 4 I/O schedulers. In NILFS2 file system, NOOP I/O scheduler has 2% faster in writing and Deadline I/O scheduler has 6% faster in reading than other I/O schedulers.

Key Words : NAND Flash Memory, I/O scheduler

* 경북대학교 컴퓨터학부, 제1저자
** 계명문화대학 컴퓨터학부, 제2저자
*** 경운대학교 컴퓨터공학과, 제3저자
**** 경운대학교 컴퓨터공학과, 제4저자
***** 경북대학교 컴퓨터학부 (e-mail:qortn29@kun.ac.kr)

1. 서 론

현재 NAND 플래시 메모리는 가전기기, 통신기기, 휴대기기 등의 다양한 기기의 저장매체로 사용되고 있다. 최근 내장형 기기에서 필요한 저장 장치에 대용량의 메모리가 요구되기 때문에 NAND 플래시 메모리가 이러한 기기의 기본 저장매체로 각광 받고 있다.

NAND Flash Memory는 비휘발성의 특징을 가지는 반도체의 저장매체로 집적도가 높고 낮은 소비전력으로 구동이 가능하다. 또한 빠른 읽기, 쓰기 성능을 보이며 온도와 충격에 대한 강한 내구성을 가지고 있기 때문에 불안정한 환경에서 동작해야 하고, 그 크기와 전원 공급이 제한된 내장형 기기에 매우 적합하다. 이러한 NAND 플래시 메모리를 효율적으로 이용하기 위해서는 NAND Flash 파일시스템을 이용하여야 된다[1].

파일시스템은 대표적으로 NILFS2, JFFS2, YAFFS2 등이 있다. NAND 플래시 메모리에서 사용되는 파일시스템은 다음과 같은 3가지 조건을 갖추어야 한다. 첫째, 덮어쓰기가 불가능한 NAND 플래시 메모리의 특성을 고려하여, 쓰기가 일어나면 파일시스템의 내용만 기록한다. 둘째, 비정상 종료에 대한 복구기능을 지원해야 한다. 셋째, 파일시스템의 데이터저장과 처리를 위한 I/O 성능이 우수해야 한다. 특히, I/O 성능은 NAND 플래시 메모리의 데이터 처리속도에 있어서 가장 중요한 조건으로서 I/O 성능을 다양하게 분석할 필요가 있다[1].

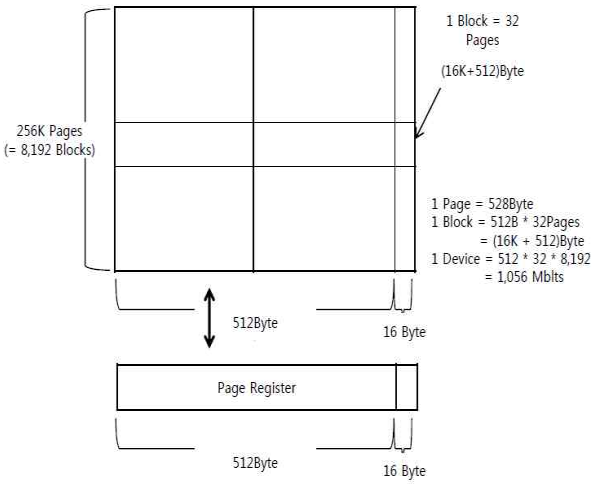
본 논문은 NAND 플래시 메모리 상에서, 각 3개의 파일시스템에 대해 4개의 I/O scheduler를 적용하여 읽기, 쓰기 시간을 비교함으로써 각 파일시스템에 대한 최적의 I/O scheduler를 제시한다.

2. 관련연구

2.1 NAND 플래시 메모리

NAND 플래시 메모리는 NOR 플래시 메모리와 마찬가지로 삭제연산의 기본단위인 블록으로 나누어진다. 블록들의 크기는 모두 동일하고, 읽기, 쓰기 연산의 기본단위인 256바이트, 512바이트, 2048바이트 크기의 페이지(Page)로 좀 더 세분화된다. 또한 각 페이지는 8바이트, 16바이트, 64바이트의 저장 영역을 추가로 가지는데, 이 잉여 영역(spare area)은 많은 응용에서 메타데이터(metadata)나 에러정정코드(error correction code, ECC)를 저장하는데 사용된다.

그림 1은 NAND 플래시 메모리의 구조를 나타낸 것이다.



<그림 1> NAND 플래시 메모리

플래시 메모리의 수명은 삭제주기(erase cycle)로 결정되는데, 평균적으로 NAND 플래시 메모리의 경우 1,000,000회의 삭제주기를 가진다. 플래시 메모리에서의 특정 블록에 지우기 작업이 집중되어 해당 블록만이 수명이 다하는 것을 방지하기 위하여 전체 블록의 삭제주기가 고루 분산되도록 관리되어야 하는데 이러한 과정을 마모도 평준화(wear leveling)라고 한다.

NAND 플래시 메모리는 출하 시 약간의 베드 블록을 가지고 있을 수 있으며 삭제연산의 횟수가 증가함에 따라 마모도가 증가하여 베드 블록이 발생할 수도 있다. 따라서 NAND 플래시 메모리를 사용하는 응용에서는 이러한 부분에 대한 처리도 필요하다[1].

2.2 파일시스템

파일시스템이란 파일의 실제 데이터와 메타데이터(파일의 위치, 크기, 소유자, 허가권 등의 파일정보)를 유지, 저장하는 체제이다. Linux 계열 파일시스템은 EXT4, UBIFS, JFFS2, YAFFS2, NILFS2 등이 있다. 파일시스템에서 중요한 요소 중 하나는 저장 매체에 대한 안정성과 효율성을 고려해야 한다.

2.2.1 YAFFS2

YAFFS2(Yet Another Flash File System 2) 파일시스

템은 체크 포인트의 기능을 도입하여 YAFFS 파일시스템의 초기화 지연에 대한 문제점을 극복했다. 체크 포인트란 로그내의 정보를 반영하기 위하여 데이터 파일을 갱신하는 트랜잭션 로그의 일정한 시점이다. 체크 포인트 기능의 단점은 정상적인 Mount 해제가 이루어지지 않을 시, NAND Flash 전체의 영역을 다시 검색하는 지연이 발생하는 것이다. Out-Place-Update 기법은 NAND 플래시 메모리의 In-Place-Update가 되지 않는 문제점을 해결 할 수 있는 장점이 있다[2]. YAFFS2 파일시스템은 갑작스러운 Power Failure나 언마운트시 체크 포인트가 저장되지 않으면 전체 영역을 스캔해야 하는 문제점이 존재한다.

2.2.2 NILFS2

NILFS2(New Implementation of a Log-structured File System 2)는 일본의 NTT(Nippon Telegraph and Telephone)에서 개발한 Log-structured 파일시스템이다[3]. NILFS2는 2007년 중반에 처음 릴리스 되었으며 NILFS2에는 가비지 컬렉션과 다수의 스냅샷을 작성하고 유지하는 기능이 포함되었다. 2009년에는 NILFS2 파일시스템이 주요 Linux kernel에 포함 되었으며 Linux의 loader convertible모듈을 설치하면 이 파일시스템을 간단히 사용할 수 있다.

NILFS2는 연속적인 스냅샷이 가능하다. NILFS2는 로그 형태로 구조화되어 있기 때문에 새 데이터는 로그의 헤드에 쓰여지게 되며 기존 데이터는 가비지 컬렉션 되지 않는 한 계속 존재한다. 기존 데이터가 계속 존재하기 때문에 원하는 시점으로 돌아가서 해당 파일시스템의 Epoch를 조사할 수 있다. 이러한 Epoch는 NILFS2에서 체크 포인트라고 불리며 파일시스템의 핵심 부분이 된다. NILFS2에서는 파일시스템의 변화가 생길 때마다 이러한 체크 포인트가 작성되지만 사용자가 강제로 체크 포인트를 작성할 수도 있다.

NILFS2는 쓰기가 순차적으로 이루어지고 이로 인해 실제 디스크의 찾기 작동이 최소화된다. 그러므로 디스크 쓰기 속도가 매우 빠르다. 일반적으로 파일시스템은 빠르게 작동하지만 가비지 컬렉션이 필요한 경우에는 성능이 저하된다.

2.2.3 JFFS2(Journaling Flash File System2)

JFFS2 파일시스템은 JFFS 파일시스템을 기반으로 Redhat Linux 운영체제에서 개발한 것으로 MTD 드라이버에 의해 NAND 플래시 메모리에 구현이 가능하다[4]. JFFS2 파일시스템은 JFFS 파일시스템의 가비지 컬렉션 문제를 극복한 임베디드 시스템을 위한 저널링 파일시스템이다.

JFFS2 파일시스템은 로그 구조 파일시스템의 원리를 플래시 파일시스템의 설계에 적용하여 구현한 예이다. 로그 구조 파일시스템은 저장 장치의 새로운 영역에 변경된 데이터를 계속 추가해 나가는 방식으로 파일시스템의 데이터 갱신 작업에 효율적이다.

JFFS2 파일시스템은 비휘발성 특성이 있으므로 데이터를 안정적으로 보호한다. JFFS2 파일시스템은 마운트 할 때 플래시 메모리 전체를 스캔해야 하기 때문에 플래시 메모리의 용량에 따라서 마운트 시간이 길어지게 된다.

2.3 Linux I/O scheduler

I/O scheduler는 block layer에 해당하며, 파일시스템으로부터 Bio Structure에 대한 Request Submit를 받아 Block I/O에 대한 동작 요청을 I/O Request Queue로 전달하는 역할을 한다. I/O Request Queue에 전달된 I/O는 Device Driver에게 동작을 수행할 것을 요청할 수 있다.

현재 Linux에서 제공하고 있는 I/O scheduler는 NOOP, Deadline, Anticipatory, CFQ 등 4가지가 있다[5, 6].

2.3.1 NOOP(No Operation) I/O scheduler

NOOP I/O scheduler는 가장 간단한 형태로 우선순위 없이 FIFO(First In First Out)로 처리된다[7]. 각 요청에서 인접한 부분들을 병합하고, 이 작업 외에는 아무 일도 하지 않는다. 탐색 시간이 없기 때문에 인접한 부분을 병합하는 것만으로도 충분히 최고의 성능을 낼 수 있다. 다중 작업을 하는 경우에는 최악의 성능을 나타낸다.

2.3.2 Deadline I/O scheduler

Deadline I/O scheduler는 리눅스 엘리베이터 I/O scheduler가 현재 요청과 인접 요청을 병합하여 처리하는 특성 때문에 떨어진 곳에서 요청이 발생되며 처리되지 못하고 대기하는 기아 현상을 방지하기 위해서 마감시한을 둔다. 읽기, 쓰기를 처리하기 위한 읽기 FIFO 요청 큐와 쓰기 FIFO 요청 큐가 있으며, 각 요청을 마감시한에 따라 정렬한 큐를 갖고 있다. 읽기 요청에 대한 마감시한은 0.5초이며, 쓰기 요청에 대한 마감시한은 5초로 설정되어 있다. Deadline I/O scheduler는 시스템에 많은

영향을 끼치는 읽기 요청을 들어오는 즉시 처리하므로 시스템 반응 속도가 빨라진다. 그러나 쓰기 요청을 버퍼에 담아 두고 읽기 요청이 끝나고 나서 처리하기 때문에 쓰기 속도가 느려진다[7].

2.3.3 Anticipatory I/O scheduler

Anticipatory I/O scheduler는 Deadline I/O scheduler와는 다른 방법을 사용한다[7]. Deadline I/O scheduler는 효율적인 I/O 처리를 위해 요청이 들어오면 즉시 해당 요청을 처리한다. 이로 인해서 여기저기 흩어져 있는 블록들에 대한 요청이 지속적으로 들어오는 상황이라면 Deadline I/O scheduler는 메모리 위치를 자주 이동해야 하기 때문에 검색 시간이 매우 큰 비중을 차지한다.

Anticipatory I/O scheduler는 I/O 처리를 위해 발생하는 요청을 즉시 처리하는 대신 처리 시간을 약간만 지연시키고, 지연 된 시간동안 들어온 요청들을 처리하는 방식이다. Anticipatory I/O scheduler는 I/O 처리를 위해, 요청을 바로 처리하는 대신에, 지연 된 시간동안 들어온 요청들을 한꺼번에 처리하는 방식이다. 이로 인해, 검색 시간이 줄어든다. 그러나 주변 요청이 없을 경우 해당 요청을 처리하기 위해서 대기시간 동안 기다려야 하는 상황이 발생한다.

2.3.4 CFQ I/O scheduler

CFQ(Complete Fair Queuing) I/O scheduler는 요청을 처리하기 위해서 큐를 만들고, 각 큐를 Round Robin 방식으로 공평하게 하나씩 방문하여 처리해 주는 방식이다[7].

한 번에 4개의 큐를 처리할 수 있다. 이 방식은 멀티미디어에 맞춰서 설계된 방식이며, 공평하게 I/O 요청을 처리할 수 있는 기회를 부여하기 때문에 데스크 탑 환경에서도 잘 동작한다. 그러나 현재 위치보다 뒤에 있는 위치를 탐색할 경우 다른 I/O scheduler보다 탐색할 수 있는 범위가 작다.

3. 실험 및 결과

3.1 성능분석

NAND 플래시 메모리 상에서 JFFS2 파일시스템에 대해 CFQ I/O scheduler를 이용하여 읽기와 쓰기 시간을 측정, NOOP I/O scheduler를 이용하여 읽기와 쓰기 시간을 측정, Anticipatory I/O scheduler를 이용하여 읽기와 쓰기 시간을 측정, Deadline I/O scheduler를 이용하여 읽기와 쓰기 시간을 측정한다. YAFFS2 파일시스템에 대해서도 동일하게 측정하며, JFFS2 파일시스템에 대해서도 동일하게 측정한다. 그러므로 각 파일시스템에 대한 I/O scheduler의 읽기, 쓰기 측정 횟수가 총 24회이다. 본 논문에서는 3개의 파일시스템에 대해 4개의 I/O scheduler를 적용하여 데이터 읽기, 쓰기 시간을 총 24회 비교함으로써 각 파일 시스템에 대한 최적의 성능을 제시한다.

3.2 실험 환경

본 논문의 실험 환경은 S5PV기반 프로세스, 512Mbyte RAM 메모리와 256Mbyte NAND Flash를 탑재한 하드웨어 상에서 테스트를 수행한다. Linux 커널은 2.6.29 버전을 사용하였으며, YAFFS2, JFFS2, NILFS2 파일시스템은 모듈 형태로 Linux상에 적재하여 테스트 한다. 성능 평가를 위한 하드웨어 사양은 <표 1>와 같다.

성능분석을 위해, 파일시스템 벤치마킹 툴은 IOZONE을 사용한다.

<표 1> 실험 환경의 하드웨어 사양

구분	사양
CPU	Samsung S5PV210 ARM Cortex A8 800MHz
Memory	Mobile DDR2 512Mbyte
Flash	256Mbyte SLC NAND Flash Memory
Ethernet	SMSC LAN9220(10/100Mbps)
USB	USB 2.0 Host/USB 2.0 OTG

IOZONE은 read, write, re-read, re-write, read backwards, read strided, fread, fwrite 등의 다양한 테스트 환경을 제공해준다. 이 실험에서는 가장 최신의 3.61 버전으로 성능 테스트를 하였으며, 오차를 최소화하기 위해 각 실험 당 10회 테스트가 수행되었다. <표 2>은 본 논문에서 실험한 테스트 조건이다.

<표 2> 테스트 조건

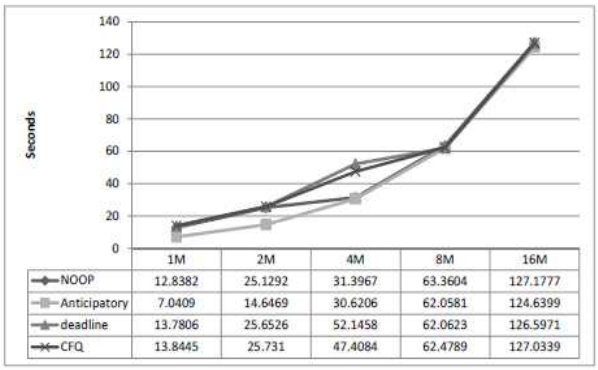
Mode	Sequential Write, Read
File System	YAFFS2, JFFS2, NILFS2
I/O Scheduler	Deadline, NOOP, CFQ, Anticipatory
Page size	32(KB)
Block size	1,2,4,8,16(MB)

3.3 실험 결과 분석

본 절에서는 YAFFS2, NILFS2, JFFS2 등의 파일시스템에 Deadline, NOOP, CFQ, Anticipatory I/O scheduler를 적용하여, 순차적인 데이터 쓰기와 읽기 성능을 비교한다. 실험은 256Mbyte의 NAND 플래시 메모리 상에서 페이지 단위 32KB에 대한 1~16MB 메모리 크기를 증가시키면서 유저 환경과 동일한 상태에서 실험을 진행하였다. 성능 측정 방식은 각 파일시스템에 대한 각I/O scheduler의 쓰기, 읽기 속도를 측정하였다.

3.2.1 순차적인 데이터 쓰기 성능 분석

그림 2는 JFFS2 파일시스템에서 NOOP, Anticipatory, Deadline, CFQ I/O scheduler들의 메모리 사용량에 따른 데이터 쓰기 시간을 분석한 결과이다.



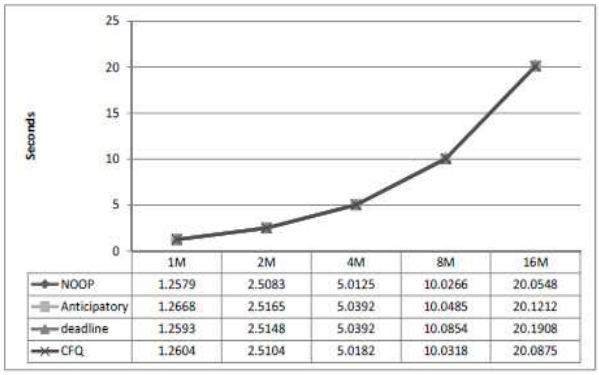
<그림 2> JFFS2에 대한 I/O scheduler의 데이터 쓰기 성능

<표 3>는 JFFS2 파일시스템에서 I/O scheduler들의 평균 데이터 쓰기 시간을 나타낸다. Anticipatory I/O scheduler는 47.49초로 다른 I/O scheduler보다 평균 데이터 쓰기 시간이 빠르다.

Anticipatory I/O scheduler는 I/O 처리 요청이 들어오면 바로 처리하지 않고, 6ms 초 정도의 짧은 시간을 대기 한 후, 이 시간 동안 들어온 I/O 처리 요청들을 함께 처리한다. 이로 인해, 다른 I/O scheduler들에 비해 빠른 성능을 나타낸다.

<표 3> JFFS2에 대한 I/O scheduler 평균 시간 측정

I/O scheduler	Average
NOOP	51.97
Anticipatory	47.79
Deadline	56.04
CFQ	55.29



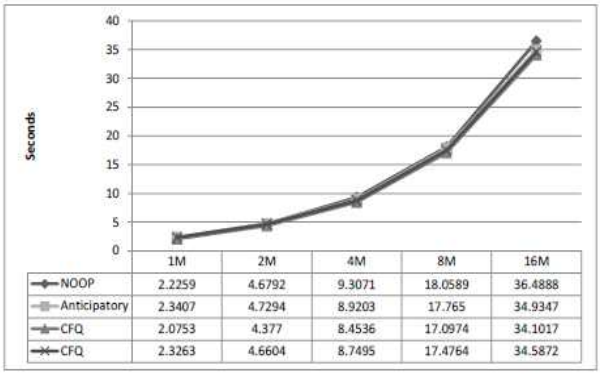
<그림 3> YAFFS2에 대한 I/O scheduler의 데이터 쓰기 성능

<표 4>는 YAFFS2 파일시스템에서 I/O scheduler의 데이터 평균 쓰기 시간을 나타낸다. NOOP I/O scheduler는 데이터 쓰기 시간이 7.76초로 다른 I/O scheduler보다 데이터 쓰기 시간이 빠르다.

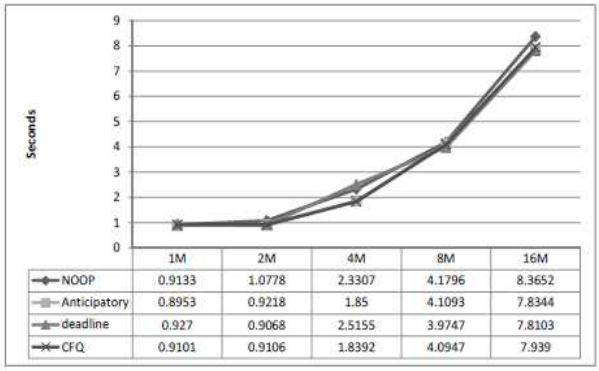
NOOP I/O scheduler는 I/O 처리 요청이 들어오면 인접한 부분들을 병합하는 작업 외에는 하지 않는 매우 단순한 기능만 수행하기 때문에 체크 포인트 기능으로 Mount 시점이 최적화 된 YAFFS2 파일시스템 상에서 메모리 사용량에 따른 데이터 쓰기 시간이 다른 I/O scheduler보다 빠르다.

<표 4> YAFFS2에 대한 I/O scheduler의 데이터 평균 시간 측정

I/O scheduler	Average(초)
NOOP	7.76
Anticipatory	7.79
Deadline	7.81
CFQ	7.77



<그림 4> NILFS2에 대한 I/O scheduler의 데이터 쓰기 성능



<그림 5> JFFS2에 대한 I/O scheduler의 데이터 읽기 성능

그림 4은 NILFS2 파일시스템에서 NOOP, Anticipatory, Deadline, CFQ I/O scheduler들의 메모리 사용량에 따른 데이터 쓰기 시간을 분석한 결과이다.

<표 5>는 NILFS2 파일시스템에서 I/O scheduler들의 평균 데이터 쓰기 시간을 나타낸다. Deadline I/O scheduler는 데이터 쓰기 시간이 13.21초로 다른 I/O scheduler보다 데이터 쓰기 시간이 더 빠른 것을 볼 수 있다. Deadline I/O scheduler는 기존 데이터가 NAND 플래시 메모리에 남아 있는 NILFS2 의 특성과 메모리에 데이터를 쓰기 하는 경우 Block 단위로 Erase 후 Page 단위로 쓰는 NAND 플래시 메모리의 특성을 읽기 요청 우선 정책을 통해 읽기 지연을 최소화하여 데이터 쓰기 시간을 다른 I/O scheduler보다 빠르다.

<표 5> NILFS2에 대한 I/O scheduler의 데이터 평균 시간 측정

I/O scheduler	Average(초)
NOOP	14.14
Anticipatory	13.61
Deadline	13.21
CFQ	13.55

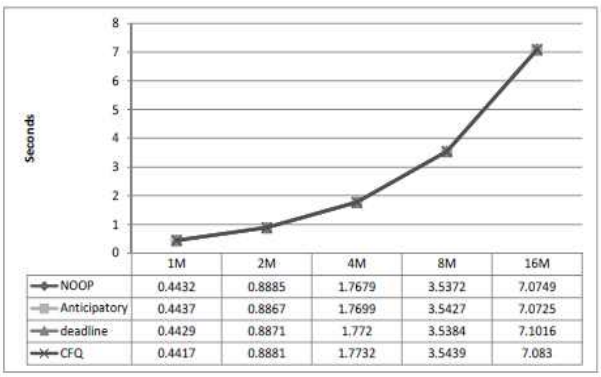
3.2.2 순차적인 데이터 읽기 성능 분석

그림 5은 JFFS2 파일시스템에서 NOOP, Anticipatory, Deadline, CFQ I/O scheduler들의 메모리 사용량에 따른 데이터 쓰기 시간을 분석한 결과이다.

<표 6>은 JFFS2 파일시스템에서 I/O scheduler의 평균 데이터 읽기 시간을 나타낸다. Anticipatory I/O scheduler는 데이터 읽기 시간이 3.11초로 다른 I/O scheduler보다 데이터 읽기 시간이 더 빠른 것을 볼 수 있다. Anticipatory I/O scheduler는 읽기 요청이 발생할 때마다 I/O scheduler는 빠르게 읽기 요청을 처리한다. 즉 JFFS2 파일시스템 상에서 Anticipatory I/O scheduler가 다른 I/O scheduler보다 데이터 읽기 시간이 빠르다.

<표 6> JFFS2에 대한 I/O scheduler의 데이터 평균 시간 측정

I/O scheduler	Average(초)
NOOP	3.34
Anticipatory	3.11
Deadline	3.22
CFQ	3.13



<그림 6> YAFFS2에 대한 I/O scheduler의 데이터 읽기 성능

그림 6은 YAFFS2 파일시스템에서 NOOP, Anticipatory, Deadline, CFQ I/O scheduler들의 메모리 사용량에 따른 데이터 읽기 시간을 분석한 결과이다.

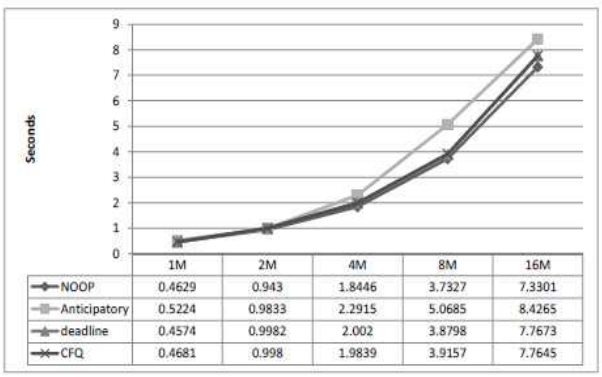
<표 7>는 YAFFS2 파일시스템에서 I/O scheduler의 평균 데이터 읽기 시간을 나타낸다. I/O scheduler들은 데이터 읽기 시간이 큰 변화를 보이지 않았다. YAFFS2는 덤프 방식(Full Dump Method)을 사용하여 체크 포인트 정보를 메타데이터에 저장하게 되어, 모든 정보를 저장하고 있는 메타데이터를 읽어 오게 되므로 읽기의 성능이 빨라지게 되기 때문에 I/O scheduler의 데이터 읽기 시간에 대한 변화 폭이 크지않다.

<표 7> YAFFS2에 대한 I/O scheduler의 데이터 평균 시간 측정

I/O scheduler	Average(초)
NOOP	2.73
Anticipatory	2.73
Deadline	2.74
CFQ	2.74

그림 7은 NILFS2 파일시스템에서 NOOP, Anticipatory, Deadline, CFQ I/O scheduler들의 메모리 사용량에 따른 데이터 읽기 시간을 분석한 결과이다.

랜덤 액세스가 가능한 디바이스에 사용하기 위해 만들어진 NOOP I/O scheduler는 NILFS2의 Log 형태로 구조화 되어 있는 특징을 이용하여 인접한 부분들을 병합하는 것만으로도 I/O scheduler 읽기 시간이 다른 I/O scheduler보다 빠르다.



<그림 7> NILFS2에 대한 I/O scheduler의 데이터 읽기 성능

<표 8>은 NILFS2 파일시스템 에서 I/O scheduler의 평균 읽기 시간을 조사하면 I/O scheduler 읽기 시간이 큰 변화를 보이지 않았다. YAFFS2는 덤프 방식(Full Dump Method)을 사용하여 체크 포인트 정보를 메타데이터에 저장하게 되어, 모든 정보를 저장하고 있는 메타데이터를 읽어 오게 되므로 읽기의 성능이 빨라지게 되기 때문에 I/O scheduler 읽기 시간이 큰 변화를 보이지 않았다.

<표 8> NILFS2에 대한 I/O scheduler의 데이터 평균 시간 측정

I/O scheduler	Average(초)
NOOP	2.86
Anticipatory	3.45
Deadline	3.01
CFQ	3.02

4. 결 론

본 논문에서는 YAFFS2, JFFS2, NILFS2 파일시스템에서 NOOP, Anticipatory, Deadline, CFQ I/O scheduler을 이용하여 쓰기, 읽기 시간을 각각 측정하였다. 분석결과, YAFFS2 파일시스템의 I/O scheduler의 읽기, 쓰기 시간이 일정하다. 즉, 초기화를 진행할 때 NAND 플래시 메모리 전체를 검색하지 않고 체크 포인트 영역을 읽어들여 파일시스템의 상태를 복원하기 때문이다. 반면, JFFS2 파일시스템 상에서 각 I/O scheduler의 읽기, 쓰기 시간을 평균적으로 보면 Anticipatory I/O scheduler의 쓰기 시간이 47.79초, 읽기 시간이 3.11초로 다른 I/O scheduler보다 쓰기 8%, 읽기 1.5% 이상 시간이 단축된다. 이는 JFFS2 파일시스템 상에서 I/O scheduler의 읽기, 쓰기가 수행될 때 에는 Anticipatory I/O scheduler가 적합하다고 볼 수 있다. 또한 NILFS2 파일시스템 상에서 각 I/O scheduler의 읽기, 쓰기 시간을 평균적 보면 Deadline I/O scheduler의 쓰기 시간이 13.21초, NOOP I/O Scheduler의 읽기 시간이 2.86초로 다른 I/O scheduler보다 쓰기 2%, 읽기 6% 정도 단축 된다. 이는 NILFS2 파일시스템 상에서 I/O scheduler의 읽기가 수행될 때는 Deadline I/O scheduler가, I/O scheduler의 쓰기가 수행 될 때는 NOOP I/O Scheduler가 적합하다고 볼 수 있다.

위와 같은 수치로 NAND 플래시 메모리를 위한 새로운 I/O scheduler 설계의 근거가 되고, 또한 NAND 플래시 메모리 전용 파일시스템을 작성하는데 있어 새로운 시각을 제시 했다고 할 수 있다. 본 연구진은 이러한 결과를 토대로 현재 새로운 Linux I/O scheduler를 설계하

고 있다.

참 고 문 헌

[1] S. O. Park, and S. J. Kim, “An efficient multimedia file system for NAND Flash Memory Storage,” IEEE Transactions on Consumer Electronics, Vol. 55, No. 2, pp. 139-145, February, 2009.

[2] YAFFS2, “http://www.yaffs.net”.

[3] NTT, New Implementation of a Log-structured File System, “http://www.nilfs.org”.

[4] D. Woodhouse, Red Hat, Inc. JFFS2: The Journaling Flash File System, version 2. available from http://sourceware.org/jffs2/, 2003.

[5] T. Ota, S. Okamoto, “Using I/O Scheduler to reduce I/O load in virtualization environments”, IEEE, Vol10.1109, pp.59-62, 2011.

[6] A. Carroll. “Linux Block I/O Scheduling”, pp. 1-11, December 22. 2007.

[7] 한동훈, "리눅스 커널 프로그래밍", pp.469-479, 한빛미디어.

[8] IOZONE, “http://www.IOZONE.org”.

[9] S. Park, K. Shen, “FIFO-A Fair, Efficient Flash I/O Scheduler”, Department of Computer Science, pp. 1-15, 2012.

[10] M. Rosenblum and J. K. Ousterhout, “The Design and Implementation of a Log-Structured File System,” ACM Transaction on Computer System, Vol. 10, No. 1, pp. 26-52, 1992.

[11] S. H. Park, T. H. Lee, and K. D. Chung, “A Flash File System Support Fast Mounting for NAND Flash Memory Based Embedded Systems”, SAMOS 2006, Lecture Notes in Computer Science, Vol. 4017, pp. 415-424, July, 2006.

이 영 석 (Yeongseok Lee)

- 학생회원
- 경운대학교 모바일공학과 학사
- 경북대학교 컴퓨터학부 석사과정
- 관심분야 : 임베디드 시스템, 파일 시스템

이 창 희 (Changhee Lee)

- 경북대학교 컴퓨터공학과 학사
- 경북대학교 컴퓨터공학과 석사
- 경북대학교 컴퓨터공학과 박사
- 계명문화대학 컴퓨터학부 교수
- 관심분야 : 임베디드 시스템, RFID, 시험구조

정 경 호 (Kyungho Chung)

- 대구대학교 컴퓨터정보공학과 학사
- 경북대학교 컴퓨터공학과 석사
- 경북대학교 컴퓨터공학과 박사
- 경운대학교 IT에너지대학 컴퓨터공학과 교수
- 관심분야 : 임베디드 시스템, RFID, 정보보호

김 용 환 (Yonghwan Kim)

- 경운대학교 컴퓨터공학과 학사
- 경북대학교 컴퓨터공학과 석사
- 경북대학교 컴퓨터공학과 박사
- 경운대학교 IT에너지대학 컴퓨터공학과 연구교수
- 관심분야 : RFID, 임베디드 시스템, 센서 네트워크

안 광 선 (Kwangseon Ahn)

- 정회원
- 연세대학교 전기공학과 학사
- 연세대학교 전자공학과 석사
- 연세대학교 전자공학과 박사
- 경북대학교 IT대학 컴퓨터공학부 교수
- 관심분야 : 임베디드 시스템 설계, RFID

논 문 접 수 일 : 2013년 02월 18일
1차수정완료일 : 2013년 03월 26일
게 재 확 정 일 : 2013년 03월 26일