JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 19, No. 1 : 136~140 Jan. 2015

# 리눅스 환경에서 파일 시스템들의 블록 쓰기 연산 성능 분석

최진오*

# Performance Analysis of Block Write Operation of File Systems on Linux Environment

**Jin-oh Choi***

Department of Embedded Software, Busan University of Foreign Studies, Busan 609-815, Korea

## 요 약

임베디드 시스템에 많이 사용되는 리눅스 환경에는 FAT, NTFS, Ext2 등 다양한 파일 시스템이 사용된다. 임베디드 시스템에 탑재된 파일 시스템은 미니 하드 디스크 또는 플래시 메모리를 미디어로 채택하고 있다. 이러한 장치에 구현되는 파일 시스템의 종류는 응용 프로그램의 성능에 많은 영향을 미친다. 동일한 미디어에서 파일시스템의 성능 요인은 블록 읽기, 블록 쓰기, 그리고 블록 해제 오버헤드이다. 이 중에서 블록 읽기와 블록 해제 성능은 파일시스템에 따라 큰 차이를 보이지 않는다. 이 논문에서는 리눅스에서 지원하는 다양한 파일시스템에서 블록 할당과 쓰기 성능을 벤치마킹한다. 다양한 실험을 통하여 얻어진 결과로부터 각 파일 시스템의 특성들을 살펴본다.

## ABSTRACT

Linux environment that is commonly used at embedded systems supports various file systems as Ext2, FAT, NTFS, etc. The file system that is equiped on the embedded system is mostly implemented on mini hard disk or flash memory. The types of the file system of the system make an effect on the performance of a application programs. The factors of file system performance on a same media are block read, block write and block free time. On these factors, block read and block free time are not so different according to the type of file systems. This paper evaluates the performance benchmark of file systems supported by linux about block allocation and write performance. The results obtained from various experiments shows the characteristics of each file system.

**키워드** : 블록 할당, 리눅스 파일 시스템, 파일시스템 성능, 임베디드 시스템

Key word : Block Allocation, Linux File System, Performance of File System, Embedded System

접수일자 : 2014. 11. 28 심사완료일자 : 2014. 12. 15 게재확정일자 : 2014. 12. 29

* **Corresponding Author** Jinoh Choi (E-mail: jochoi@bufs.ac.kr, Tel:+82-51-509-6245)
Department of Embedded Software, BusanUniversity of Foreign Studies, Busan 609-815

Open Access http://dx.doi.org/10.6109/jkiice.2015.19.1.136 print ISSN: 2234-4772 online ISSN: 2288-4165

## Ⅰ. 서 론

임베디드 시스템(Embedded System)의 운영체제는 freeRTOS, RT Linux, Android, MS-Windows, VxWorks 등이 일반적으로 사용되고 있는데, 이중에서 리눅스의 사용 비율이 가장 많이 증가되고 있다[1]. 리눅스 운영체제 환경은 i-node 기반의 가상 파일 시스템을 이용하여 다양한 파일 시스템들을 지원한다[2]. 리눅스 환경의 임베디드 시스템에서 파일 시스템을 사용하는 응용 프로그램들은 CPU와 주 메모리에 비해 느린 속도의 파일 시스템 성능 때문에 전체 성능이 파일 시스템에 의해 영향을 받는다. 리눅스환경의 임베디드 시스템에서 파일 시스템 구축 미디어로는 광학 CD-ROM, 미니 하드 디스크, 플래시 메모리, SSD(Solid State Drive) 등이 활용된다. 특히 최근 SSD와 같이 고속의 플래시 메모리 기반 파일시스템 구축이 일반화되면서 보조 기억장치에 의존하는 임베디드 시스템이 늘어나고 있다.

리눅스에서 지원하는 파일 시스템은 FAT-16, FAT-32, NTFS, Ext2, Ext3, Ext4 등이다. 각 파일 시스템은 구조와 메커니즘 특성에 따라 장단점이 존재한다. 즉, 파일 시스템의 종류에 따라 디스크에서 데이터를 검색하는 성능, 블록을 할당받아 데이터를 기록하는 성능 그리고 파일을 삭제하는 성능이 차이를 보인다.

파일 시스템들의 삭제(블록 해제) 성능은 크게 다르지 않다. 대부분 파일시스템의 메타 데이터(meta data) 영역에 삭제 표시만 한다[3]. 그리고 파일 시스템에서 검색 성능(블록 읽기)과 기록 성능(블록 할당 & 쓰기) 중에서는 기록 성능이 월등히 낮다. SSD의 경우 4배 가까이 차이가 난다[4]. 따라서 블록 할당과 쓰기 성능이 전체 파일 시스템의 성능에 가장 큰 영향을 미친다고 볼 수 있다.

이 논문에서는 임베디드 시스템을 위한 리눅스 환경에서 파일 시스템의 종류에 따라 파일 시스템 성능에 가장 큰 영향을 미치는 블록 할당과 쓰기 성능에 대하여 실험하고 그 특성과 결과를 분석한다.

## Ⅱ. 관련 연구

기존 파일 시스템 중 FAT(File Allocation Table) 파일 시스템은 단순한 구조를 가지고 있으며 FAT 영역에 별도로 파일의 데이터 블록 리스트를 저장하고 있다. 따라서 FAT 내용만으로 모든 파일의 배치 구조를 알 수 있는 장점이 있다. 특히 블록 할당 시 FAT만 탐색함으로써 빈 블록을 빨리 찾을 수 있다는 장점이 있다. 그림 1은 FAT 파일 시스템의 개념도이다[3].


**그림 1.** FAT 파일 시스템
**Fig. 1** FAT File System

NTFS(New Technology File System)는 FAT의 한계를 극복하고 서버용으로 개발되어 FAT보다 복잡한 구조를 가지고 있으며 MFT(Master File Table)로 모든 개체를 저장하며 내부에 다양한 속성(attribute)들을 가지고 있다. 블록 할당에는 내장 MFT 파일 정보를 이용한다. 그림 2는 NTFS의 디스크 구조이다[3].

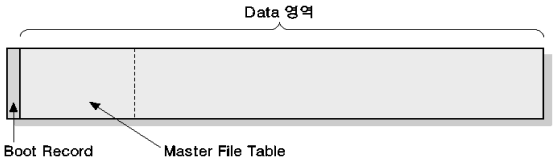


**그림 2.** NTFS 파일 시스템
**Fig. 2** NTFS File System

Ext(Extended) 파일 시스템은 파일 시스템을 여러 블록 그룹으로 쪼개어 별도 관리하여 다소 복잡한 구조를 가진다. 각 블록 그룹은 i-node로 파일 정보를 저장하는 방법을 채택하며 Bitmap들을 도입하여 빠른 접근을 지원한다. 특히 블록 할당 시 블록 Bitmap을 이용하여 빈 블록을 빨리 탐색할 수 있다는 장점이 있다. 그림 3은 Ext 파일 시스템의 디스크 구조이다[3].

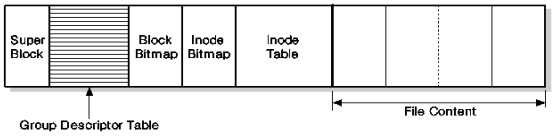


**그림 3.** Ext2 파일 시스템
**Fig. 3** Ext2 File System

리눅스 환경에서 파일 시스템의 블록 할당과 쓰기 성능에 대한 연구로 [5]에서는 Bonnie++라는 벤치마킹 도구를 이용하여 파일시스템들의 성능 분석을 하여 결과를 보이고 있다. [6]에서는 신뢰성 보장을 위한 쉐도잉 오버헤드를 고려한 쓰기 성능 분석 연구를 수행하였다. 기타 다양한 연구에서 리눅스 파일 시스템의 성능, 특히 SSD에서의 성능에 대한 연구를 벤치마킹 수준에서 수행한 결과를 보이고 있다. 그러나 이 연구에서는 각 파일 시스템의 블록 할당과 쓰기 성능을 보다 자세히 살펴보고 성능 특성을 논의하고자 한다. 블록 할당은 빈 파일 시스템과 복잡하게 할당 블록과 빈 블록이 섞여 있는 환경으로 나누어 실험한다.

## Ⅲ. 파일 시스템의 성능 비교

### 3.1. 파일 시스템의 성능 요인

파일 시스템의 연산은 블록 읽기, 블록 할당 및 쓰기, 그리고 블록 해제(파일 삭제)로 나뉜다. 읽기와 쓰기 연산은 다시 순차 접근과 임의 접근으로 나뉜다. 따라서 파일 시스템의 성능은 이 연산들의 성능으로 비교할 수 있을 것이다. 여기서 블록 읽기 연산은 가장 빠른 연산으로서 파일 시스템 별 성능차이는 크지 않다. 블록 해제 연산도 단순히 삭제 표시만 기록하므로 파일 시스템 별 성능 차이는 미미할 것이다. 따라서 블록 할당과 쓰기 연산이 전체 파일 시스템의 성능을 결정한다.

### 3.2. 파일 시스템별 블록 할당 메커니즘 비교

FAT 파일 시스템은 빈 디렉터리 엔트리(directory entry)를 할당 받고 FAT 정보를 참조하여 빈 블록들을 할당 받는다. 할당 받은 블록들은 FAT에 체인(chain)으로 연결 정보를 기록한다.

NTFS는 빈 MFT 엔트리를 하나 할당 받고 빈 블록들은 $Bitmap 파일을 참조하여 할당 받는다. 할당받은 블록들은 $DATA 속성에 저장한다.

Ext 파일 시스템은 새로운 파일을 생성하기 위해 먼저 빈 i-node를 i-node Bitmap에서 찾아 할당 받고 빈 블록들을 블록 Bitmap에서 찾아 할당 받는 과정을 거친다. 할당받은 블록들은 i-node에 목록을 저장한다. 마지막으로 파일 이름과 i-node 번호를 저장할 디렉터리 엔트리(directory entry)를 할당받아 저장한다.

### 3.3. 실험 환경

실험에서 사용된 시스템 환경은 다음과 같다.

- 리눅스 OS : Fedora 11, Kernel 2.6.29.4
- 플래시 메모리 : LG UJ1 2GB
- CPU : Intel Core Duo, 2.83GH
- Read/Write Block Size : 4Kb
- Read/Write File Size : Random Size

실험값 측정은 각각 1,000회 반복하여 측정값을 평균하였다. 파일 시스템의 블록 할당 성능은 자체 접근 메커니즘에 따라 차이가 나겠지만 특정 시점의 파일 시스템 상태에 따라 다른 특성을 보일 수도 있다. 따라서 실험을 가능한 두 가지 상태로 나누어 실시한다. 첫째, 파일 시스템이 비어 있는 경우이다. 순수하게 블록을 순차 할당받아 지속적으로 파일을 기록하는 시나리오이다. 둘째, 파일 시스템이 중간 중간에 빈 공간을 가지는 시나리오이다. 약 50%의 빈 공간이 랜덤하게 남아있는 경우이다. 빈 공간의 크기도 랜덤하다.

## Ⅳ. 실험 결과와 분석

먼저 블록 읽기 성능은 그림 4와 같다. 평균 0.008초(순차)/0.01초(임의)의 시간을 보였다.

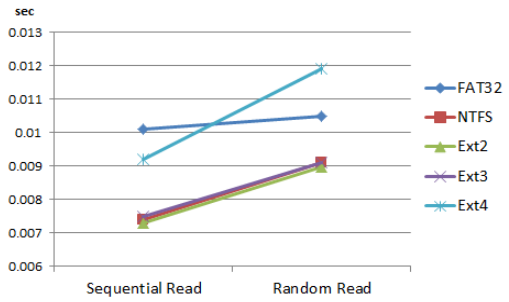


**그림 4.** 블록 읽기 성능
**Fig. 4** Block Read Performance

파일 삭제 성능을 분석한 결과는 그림 5와 같다. 평균 1개의 파일을 삭제하는 데 0.094초의 시간이 걸렸다. 블록 쓰기 성능은 그림 6과 같다. 평균 1.03초(순차)/2.13초(임의)의 시간을 보였다.

그림 7은 3종류의 파일 시스템 연산을 비교한 결과이다. 블록 읽기와 파일 삭제는 블록 할당과 쓰기에 비해 성능 차이가 아주 크고 파일 시스템 간 시간 차이도 미미하다. 쓰기 연산은 읽기보다 130여배, 삭제보다 20여배 느렸다.

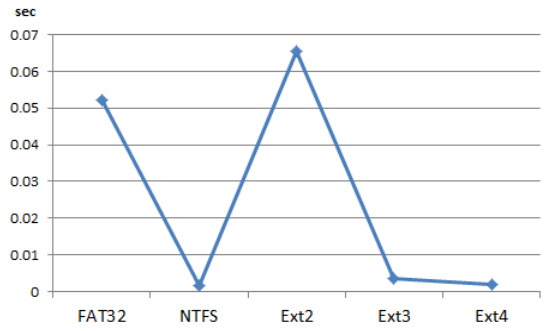


**그림 5.** 파일 삭제 성능
**Fig. 5** File Delete Performance

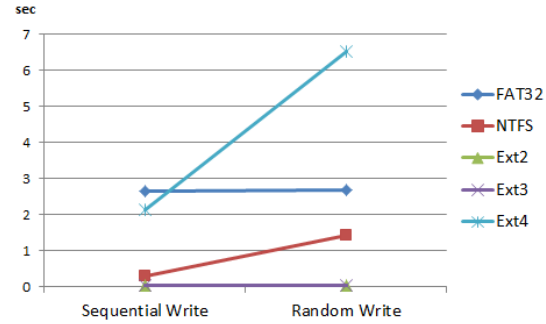


**그림 6.** 블록 쓰기 성능
**Fig. 6** Block Write Performance

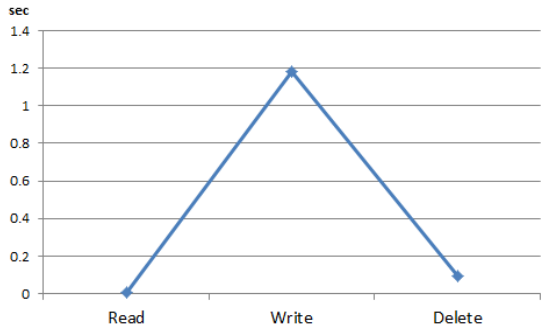


**그림 7.** 파일 시스템 연산 성능 비교
**Fig. 7** Performance Comparison of File System Operations

따라서 블록 할당 및 쓰기 연산에 대한 보다 상세한 분석을 진행하였다. 실험 환경은 먼저 빈 파일 시스템에서 각 파일 시스템별로 블록 할당 성능을 실험하였다. 그 뒤 랜덤하게 약 50%의 빈 공간이 있는 파일 시스템에서 블록 할당 성능을 실험하였다. 그림 8은 실험 결과이다. 이 실험에서 빈 파일 시스템과 랜덤한 파일 시스템에서의 성능은 파일 시스템의 종류에 관계없이 서로 비례하는 것으로 보였다. 그리고 단순한 구조의 FAT32가 가장 성능이 우수한 것으로 보였다.

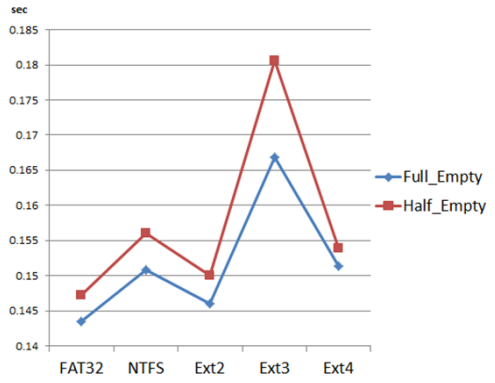


**그림 8.** 블록 할당 성능
**Fig. 8** Block Allocation Performance

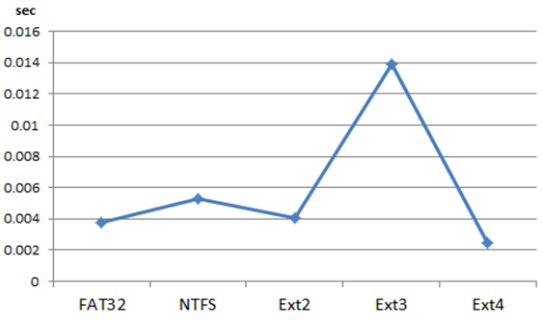


**그림 9.** 빈 FS와 50% 랜덤 FS에서의 성능 비교
**Fig. 9** Performance Comparison of Empty and 50% Random File System

FAT32가 가장 우수한 것은 FAT 조작 외의 다른 오버헤드가 없기 때문으로 판단된다. NTFS의 경우 MFT (Master File Table) 생성 후 Non-Resident Attribure를 생성하는 오버헤드가 제법 큰 것으로 보인다. Ext2의 경우 i-node Bitmap과 Block Bitmap을 사용하므로 빈

블록 할당이 효율적인 것으로 보인다. Ext3는 저널링(Journaling)과 단편(Fragment)의 최소화에 대한 오버헤드로 좋은 성능이 나오지 않았다. Ext4에서는 이러한 문제가 많이 완화되었음을 알 수 있다.

그림 9는 각 파일 시스템의 빈 파일 시스템과 50% 랜덤하게 채워진 파일 시스템에서의 블록 할당 성능 차이를 수치화한 실험결과이다. 여기서는 Ext4가 가장 차이가 적었다. 이것은 첫째, Ext4는 단편의 관리에 의한 효율적인 저장이 가능하기 때문이며 둘째, Delayed Allocation 기법에 의한 효율 때문으로 보인다.

## Ⅴ. 결 론

이 논문에서는 임베디드 시스템을 위한 리눅스 환경에서 파일 시스템의 블록 할당 성능에 대하여 고찰하였다. 관찰 대상은 가장 많이 사용되는 FAT, NTFS, 그리고 Ext 계열 파일 시스템으로서 빈 파일시스템과 랜덤하게 분포된 파일 시스템 시나리오에서 블록 할당에 대한 성능을 비교 분석하였다.

실험 및 분석 결과 다음과 같은 결론을 도출하였다. 첫째, 파일 시스템에서 연산에 따른 성능 결정 요인은 블록 할당 시간이라는 것이다. 다른 연산보다 20에서 130여배 차이가 났다. 둘째, 단편화 및 저널링 등 복잡한 다른 요소를 모두 배제하고 단순한 블록 할당 시간 비교에서는 간단한 구조와 오버헤드가 거의 없는 FAT32가 가장 우수하였다. 그 다음으로 Ext2와 Ext4가 우수하였다. 넷째, 빈 파일 시스템과 50% 랜덤한 파일 시스템에서의 블록 할당 성능은 서로 비례하였다. 하지만 Ext4에서 그 성능 차이가 가장 적어서 50% 랜덤한 환경에서 우수한 것으로 보였다.

향후 실험에 의한 정확한 결과 도출로 추론과 비교 분석이 필요하며 보다 다양한 성능 비교 인자를 개발하여 비교 분석하는 실험과 연구가 필요하다. 또한 표준 벤치마크 프로그램에 의한 성능 비교도 필요하다. 그리고 분석한 결과를 바탕으로 쓰기 연산의 성능 향상 기법을 개발할 수 있을 것으로 기대한다.


### 감사의 글

이 논문은 2014년도 부산외국어대학교 학술연구조성비에 의해 연구되었음


## REFERENCES


[1] http://www.itdaily.kr/atl/view.asp?a_id= 44297

[2] S. J. Back, J. M. Choi, "Linux Kernel Internal," Kyohaksa, 2008.

[3] J. S. Jung, W. Y. Jung, "File System Principle and Exercise for Embedded Developers," HanbitMedia, 2007.

[4] http://grwings.com/435, "Comparison of Read and Write Speed of HDD and SSD“

[5] H. J. Park, "A Study on the Performance Evaluation of File Systems using kernel-based Benchmarking Tool in the Linux," *in Journal of KIIT,* vol. 10, no. 12, pp. 119-125, Dec. 2012.

[6] S. T. Ryu, H. J. Kim, H. S. Han, "File Write Performance Analysis on Reliability-guaranteed Techniques for Non-Volatile Memory File System," *in Journal of KISE,* vol. 19, no. 11, pp. 591-595, Nov. 2013



**최진오(Jin-Oh Choi)**

1991년 부산대학교 컴퓨터공학과 공학사
1995년 부산대학교 컴퓨터공학과 공학석사
2000년 부산대학교 컴퓨터공학과 공학박사
2000년 ~ 부산외국어대학교 임베디드소프트웨어학과 교수
※관심분야 : 모바일 GIS, USN, Embedded System