NAND 플래쉬 파일 시스템의 초기화 및 크래쉬 복구 기법의 성능 모델링

(Performance modeling of initialization and crash recovery mechanism of NAND flash file system)

김 태 경, 박 현 찬, 유 혁*
(Tae-Kyung Kim, Hyun-Chan Park, Hyuck Yoo)

Abstract : The log-structured NAND flash file systems has a scalability problem that the initialization cost at mount time is increased as a capacity of flash memory media. To solve this problem, new initialization techniques such as Snapshot and Log-based method were presented. However, these techniques cannot prevent the increment of crash recovery cost when a file system was not cleanly unmount. In this paper, we analyze the initialization and crash recovery mechanism of JFFS, YAFFS, CFFS, Snapshot, and Log-based method. We propose the performance model of an each of those techniques. By analyzing the performance models, we could find out that crash recovery cost of existing flash file system is increased by a capacity of flash memory media. Therefore we conclude that we need a new NAND flash file system model which has not influenced by a capacity of flash memory media.

Keywords : 플래시 파일 시스템, NAND 플래쉬 메모리, 초기화, 크래쉬 복구, 로그 파일 시스템

Ⅰ. 서 론

임베디드 시스템에 주로 사용되는 플래쉬 메모리(flash memory)의 용량은 1년에 2배에 가까운 속도로 발전해왔다. 그러나 플래쉬 메모리 발전 속도와는 달리, 플래시 파일 시스템들은 아직 충분한 확장성을 제공하지 못하고 있다. 특히 파일 시스템의 초기화를 수행하는 기법이 플래쉬 메모리 용량에 비례함으로써 큰 용량의 메모리가 장착된 장치일수록 초기화 시간이 매우 지연되는 경향을 보이고 있다. 이러한 현상은 JFFS나 YAFFS 같은 슈퍼블록을 사용하지 않는 로그 기반 구조(Log-Structured)[1]의 플래쉬 파일 시스템들에서 두드러지게 나타나고 있다.

이러한 문제를 해결하기 위해 기존 하드디스크 파일 시스템에서 사용되는 슈퍼블록의 개념을 이용한 접근을 통해 설계된 스냅샷 기법[2], 로그 기반 기법(Log Based Method)[3], CFFS(Core Flash File System)[4] 등이 제시 되었다. 이러한 기법들은 새로운 시도를 통해 초기화 속도의 향상을 이루어 낼 수 있었다. 하지만 초기화 속도 향상을 이루기 위한 시스템은 많이 제시 되었지만 아직까지 비정상 종료가 일어난 크래쉬(crash) 상황에서의 초기화 속도 문제를 해결한 시스템은 제시되지 않았다.

플래쉬 메모리가 주로 사용되는 임베디드 시스템 환경은 전원이 유실될 우려가 다른 컴퓨팅 환경에 비해 매우 높다. 배터리를 주 전력으로 사용하는 핸드폰, PDA 등의 이동형 장비의 경우에는 전력이 유실될 확률이 더 크다. 때문에 크래쉬 상황에서의 초기화 속도도 역시 보장 되는 플래쉬 메모리 파일 시스템은 필수적일 것이다.

우리는 본 논문에서는 JFFS, YAFFS, CFFS, 스냅샷 기법, 로그 기반 기법의 초기화 성능, 크래쉬 복구 성능을 수학적으로 모델링 하였다. 분석된 모델을 통해 이러한 기법들의 초기화, 크래쉬 복구 시간이 플래시 메모리의 사용/비사용 용량에 비례함을 보이고 사용/비사용 용량에 비례하지 않는 새로운 플래쉬 파일 시스템이 필요하다는 결론을 도출하였다.

* 교신저자(Corresponding Author)
김태경, 박현찬, 유혁 : 고려대학교 컴퓨터학과

Ⅱ. 플래쉬 파일 시스템의 초기화 기법

1. 기존 플래쉬 파일 시스템의 초기화

기존의 JFFS, YAFFS는 기본적으로 [1]과 같은 로그 구조 기반의 파일 시스템을 채용하고 있다. 로그 구조 기반의 파일 시스템은 슈퍼블록과 같은 약속된 위치에 메타데이터를 저장하지 않는다는 특징을 갖고 있다. 그렇기 때문에 파일 시스템을 시작할 때 이 메타데이터들의 위치를 파악하고 메인 메모리에 정보를 정리하여 저장하는 초기화라는 작업이 필요한 것이다.

그림 1.은 JFFS의 초기화 알고리즘을 보이고 있다. JFFS는 메타데이터를 저장할 때 파일 데이터와 합쳐서 페이지 단위로 저장을 한다. 그렇기 때문에 초기화 시에 메타데이터의 정보를 얻어내기 위해서는 모든 페이지의 모든 내용의 읽기 작업이 필요하다. 이러한 작업을 그림 1.에서는 전체페이지 읽기(Full page read)라 한다.

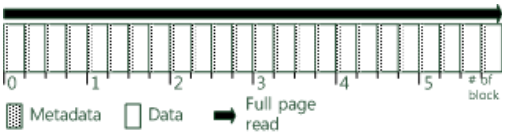

그림 1. JFFS의 초기화 알고리즘

YAFFS는 JFFS가 모든 페이지에 대해서 데이터를 읽어야 한다는 초기화 알고리즘의 단점을 극복하기 위해 파일의 메타데이터인 아이노드를 한 페이지에 저장하는 방법을 사용하고 있다. YAFFS는 여분 공간(spare area)의 ChunkID를 통해 메타데이터와 파일 데이터의 구분을 할 수 있다. 그렇기 때문에 JFFS와는 달리 메타데이터가 있는 페이지가 아닌 경우에는 여분 공간의 읽기만 하고 지나 갈 수 있어 효율적인 초기화가 가능하다. 그림 2.는 YAFFS의 초기화 알고리즘을 보이고 있다.

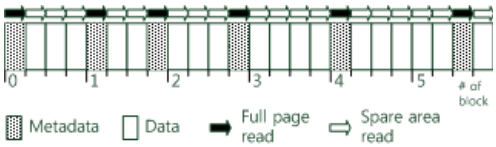

그림 2. YAFFS의 초기화 알고리즘

JFFS에 비해 개선되었기는 하지만 전체 여분 공간을 읽어야 하는 이 알고리즘은 플래쉬 메모리 용량이 증가함에 따라 수행시간도 선형적으로 증가할 것을 쉽게 예측할 수 있다. 실제로 여러 실험결과 [2,3]에서도 이러한 점을 파악할 수 있다.

2. 슈퍼블록을 이용한 플래쉬 파일 시스템의 초기화

JFFS와 YAFFS의 초기화 성능이 저하되는 가장 큰 이유는 하드디스크 파일 시스템과는 달리 슈퍼블록이 없기 때문에 메타데이터의 위치나 정보를 알 수 있는 방법이 없기 때문이다. 그러한 단점을 해결하기 위해 슈퍼블록의 개념을 이용한 스냅샷 기법, 로그 기반 기법, CFFS 파일 시스템 등이 제안 되었다.

각 기법 별로 슈퍼블록의 명칭을 스냅샷 기법은 스냅샷(Snapshot), 로그 기반 기법은 로그 세그먼트 디렉토리(Log-Segment Diretories), CFFS 파일 시스템에서는 아이노드 맵 블록(i-node map block)이라는 용어로 사용하지만 본 논문에서는 슈퍼블록으로 용어를 통일한다. 슈퍼블록을 이용한 플래쉬 파일 시스템은 정상적인 종료가 이루어졌다면 초기화시 플래시 메모리를 스캔하는 작업이 없기 때문에 기존의 초기화 성능이 플래쉬 메모리 용량이 증가함에 따라 저하되는 문제를 해결할 수 있다. 그림 3.은 슈퍼블록을 이용한 플래쉬 파일 시스템의 초기화 알고리즘을 보이고 있다.

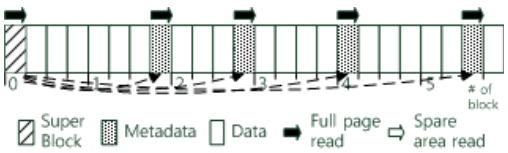

그림 3. 슈퍼블록을 이용한 초기화 알고리즘

하지만 슈퍼블록을 사용하는 파일 시스템은 크래쉬가 일어난 경우에는 슈퍼블록을 활용 할 수 없게 된다. 그러면 YAFFS와 유사하게 플래시 메모리 전체를 스캔하는 작업이 필요하게 되므로 초기화 성능이 플래쉬 메모리 용량이 증가함에 따라 수행시간도 선형적으로 증가할 것임을 알 수 있다.

Ⅲ. 모델링을 통한 성능 분석

우리는 위에서 살펴본 각 파일 시스템의 초기화 기법과 크래쉬 복구 기법의 성능을 알아보기 위해 수학적으로 동작을 모델링을 하고 그 결과를 분석

하였다.

모델에서 T_d와 T_s는 각각 한 페이지 내의 데이터 공간과 여분 공간을 읽는데 걸리는 시간을 의미한다. N_f는 파일의 개수, B_f는 사용되지 않은 블록의 수, P_{ave}는 각 파일에 포함된 평균 페이지 개수를 의미한다.

1. JFFS의 초기화 성능 모델링

JFFS는 그림 1.과 같은 알고리즘을 갖고 있고, 비정상 종료 시에도 같은 성능을 보인다.

초기화 시간 T_j를 분석하여 (1)과 같이 표현하였다. N_p는 메모리내의 "사용된" 페이지 개수이다.

$$T_j = B_f T_s + N_p(T_d + T_s) \quad (1)$$

JFFS는 모든 페이지의 데이터를 읽어 메타데이터를 읽어와 초기화 작업을 하므로 전체 플래쉬 메모리 공간에 대해 읽기 수행을 한다.

2. YAFFS의 초기화 성능 모델링

YAFFS는 그림 2.와 같은 알고리즘을 갖고 있고, 비정상 종료 시에도 같은 성능을 보인다.

초기화 시간 T_y를 분석하여 (2)와 같이 표현하였다.

$$T_y = B_f T_s + N_f(T_d + T_s) + N_f P_{ave} T_s \quad (2)$$

YAFFS는 블록의 첫 번째 페이지 여분 공간을 읽어 해당 블록이 비었는지 여부를 검사하여 내용이 있다면 페이지들을 검사한다. 페이지들의 여분공간을 검사하여 메타데이터이면 데이터들을 읽고 파일 데이터라면 여분공간만 읽고 지나간다. YAFFS의 분석결과를 보면 T_d가 T_s에 비해 많은 시간이 필요하다는 점을 생각해 본다면 초기화 성능에 작용하는 가장 큰 요소는 N_f인 것을 알 수 있다. 그리고 B_f등의 영향도 받는 모습을 보인다. 따라서 플래쉬 메모리의 용량이 커질수록 전체 수행 시간이 크게 증가할 것으로 예상된다.

3. 슈퍼블록을 이용한 플래쉬 파일 시스템의 초기화 성능 모델링

슈퍼블록을 이용한 플래쉬 파일 시스템의 초기화 성능은 약간의 차이는 보이기는 하지만 슈퍼블록을 사용한다는 개념이 공통점이기 때문에 큰 차이를 보이지 않는다.

초기화 시간 T_{sb}를 분석하여 (3)과 같이 표현하였다. P_{sb}는 슈퍼블록을 표현하는데 필요한 페이지의 수이다.

$$T_{sb} = (P_{sb} + N_f)(T_d + T_s) \quad (3)$$

슈퍼블록을 이용한 플래쉬 파일 시스템은 슈퍼블록 공간의 데이터를 읽어 메모리의 여러 장소에 분포되어있는 메타데이터들의 위치를 찾아 메타데이터의 데이터를 읽는다. 따라서 슈퍼블록과 메타데이터블록의 전체 페이지에 대해 읽기 수행을 한다.

4. 슈퍼블록을 이용한 플래쉬 파일 시스템의 크래쉬 복구 성능 모델링

슈퍼블록을 이용한 플래쉬 파일 시스템의 크래쉬 복구과정은 크게 슈퍼블록의 유효성 검사, 메모리의 스캔 2개의 과정으로 나눠질 수 있다. P_{sb}는 스냅샷을 이루고 있는 페이지의 수이고 $P_{sb}T_d$는 유효성 검사를 과정의 모델이다.

4.1 스냅샷 기법의 크래쉬 복구 성능

크래쉬 복구 시간 T_s를 분석하여 (4)와 같이 표현 하였다. T_y는 YAFFS의 초기화 성능이다.

$$T_s = P_{sb}(T_d + T_s) + T_y \quad (4)$$

스냅샷 기법은 구조적으로 YAFFS의 시스템을 기반으로 하기 때문에 YAFFS의 스캔 과정과 같을 것이다. 그렇기 때문에 메모리의 용량과 파일 수에 비례하여 크래쉬 복구 시간이 늘어남을 알 수 있다.

4.2 로그 기반 기법의 복구 성능

로그 기반 기법의 크래쉬 복구 시간 T_l를 분석하여 (5)와 같이 표현 하였다. 여기서 L은 로그의 전체 개수를, N_{lf}는 크래쉬로 인해 유실된 파일의 개수, P_{lf}는 유실된 데이터 페이지의 수이다.

$$T_l = B_f T_s + L(T_d + T_s) + N_{lf}(T_d + T_s) + P_{lf} T_s \quad (5)$$

로그 기반 기법이 제시된 논문 [3]에는 슈퍼블록의 유효성 검사 기법에 대한 기술이 부족하기 때문에 정확한 모델링이 불가능하다. 그러므로 유효성 검사에 대한 고려는 이 모델에서는 제외하였다.

로그 기반 기법은 파일에 일어나는 쓰기/지움 연산을 모두 로그로 기록하기 때문에 YAFFS와

다르게 크래쉬 복구과정에서 가장 최근의 로그 정보를 활용할 수 있다. 그러므로 우선 사용되지 않는 블록은 한 번의 여분 공간 읽기만 수행한 후 생략하고, 유실된 파일에 대해 아이노드와 데이터 페이지를 구분하여 읽어들인다. 이 분석 결과를 보면 로그를 읽는데 소요되는 시간이 추가되었지만 그로 인해 전체 파일의 개수와 비례하는 읽기 시간이 없어지고 유실된 아이노드와 데이터에 대해서만 읽기를 수행하면 된다.

4.3 CFFS의 복구 성능

CFFS의 크래쉬 복구 시간 T_c를 분석하여 (6)과 같이 표현 하였다. P_{imb}는 i-node map block의 페이지 수이고, B_{tot}는 메모리의 총 블록 수이다.

$$T_c = P_{sb}T_d + B_{tot}T_s + N_f T_d \quad (6)$$

CFFS는 YAFFS와는 다르게 플래쉬 블록들을 inode-stored 블록, data 블록, free 블록 3가지로 분류하고 있다.[4] 이렇게 함으로써 크래쉬 복구시 스캔 과정에 페이지단위로 스페어 영역을 읽어 메타데이터 여부를 확인하던 작업을 블록단위로 읽어서 확인이 가능하기 때문에 크래쉬 복구 동작 시 성능상의 이점을 얻을 수 있다. 하지만 CFFS 역시 B_{tot}만큼 여분공간을 읽어야하고 N_f만큼 데이터를 읽기 때문에 메모리의 용량과 파일의 수에 비례하여 크래쉬 복구 시간이 걸리는 것을 알 수 있다.

IV. 결 론

본 논문에서 우리는 NAND 플래쉬 파일 시스템의 초기화 시간, 크래쉬 복구 시간이 플래쉬 메모리의 용량에 비례하여 증가하는 문제를 보이기 위해 모델링을 하였다. 모델링 결과 그로 인한 영향이 실제로 크게 나타날 수 있음을 보였다. 우리는 이를 통하여 초기화나 크래쉬 복구 과정이 플래쉬 메모리의 용량, 파일의 수, 사용 용량 등에 크게 영향을 받지 않는 새로운 플래쉬 파일 시스템이 필요하다는 결론을 도출하였다.

참 고 문 헌

[1] Rosenblum, M. and J.K. Ousterhout, "The design and implementation of a log-structured file system," ACM Transactions on Computer Systems (TOCS), 1992. 10(1): p. 26-52..

[2] K.S. Yim, et al., "A fast start-up technique for flash memory based computing systems," Symposium on Applied Computing: Proceedings of the 2005 ACM Symposium on Applied computing, vol. 13, no. 17, pp. 843-849, 2005.

[3] Wu, C.H., T.W. Kuo, and L.P. Chang, "The Design of efficient initialization and crash recovery for log-based file systems over flash memory," ACM Transactions on Storage (TOS), 2006. 2(4): p. 449-467.

[4] S.H. Lim and K.H. Park, "An Efficient NAND Flash File System for Flash Memory Storage," IEEE TRANSACTIONS ON COMPUTERS, pp. 906-912, 2006.

저 자 소 개

김 태 경(Tae-Kyung Kim)

2007년 8월 : 단국대학교 정보컴퓨터학부 학사
2008년 3월~현재 : 고려대학교 컴퓨터학과 석사과정
관심분야 : 임베디드 소프트웨어, 플래쉬 파일 시스템
Email : tkkim@os.korea.ac.kr

박 현 찬(Hyun-Chan Park)

2004년 2월 : 고려대학교 컴퓨터학과 학사
2004년 3월~현재 : 고려대학교 컴퓨터학과 석박통합과정
관심분야 : 임베디드 소프트웨어, 플래쉬 파일 시스템
Email : hcpark@os.korea.ac.kr

유 혁(Hyuck Yoo)

1982년 2월 : 서울대학교 전자공학과 학사
1986년 8월 : Master of Computer Science in University of Michigan
1990년 8월 : Ph.D of Computer Science in University of Michigan
1990년~1995년 : Sun Microsystems Lab. Researcher
1995년~현재 : 고려대학교 컴퓨터학과 교수
관심분야 : 임베디드 소프트웨어, 플래쉬 파일 시스템
Email : hxy@os.korea.ac.kr