RFFS 파일 시스템을 위한 가비지 컬렉션 정책

이상기° 이태훈 정기동
부산대학교 정보컴퓨터공학과
{bruce94°, withsoul, kdjung}@pusan.ac.kr

The Garbage Collection Policy for The RFFS File System

Sanggi Lee, Taehoon Lee, Kidong Chung
Dept of Computer Engineering, Pusan National University

요 약

최근 임베디드 시스템 개발이 활발히 진행되면서 비휘발성 메모리 요구가 커지고 있다. 이에 휴대가 용이하고, 접근시간이 빠르고, 전력소비가 적은 플래시 메모리가 많이 사용되고 있으나 상대적으로 느린 지움 시간과 지움 횟수의 한계, 플래시 메모리의 대용량화로 인한 느린 마운트 속도, 예상치 못한 파워차단으로 인한 데이터 손실 등 극복해야할 문제점이 있다. 본 논문에서는 이러한 문제점을 해결하기 위해 개발된 RFFS 파일 시스템에서 연산을 줄일 수 있는 GC(Garbage Collection) 수행 방법을 제안한다. RFFS 파일 시스템을 분석하고, 예상되는 연산시간을 토대로한 GC 수행 부하를 줄이는 방법을 제안한다.

1. 서 론

지능형 가전제품과 임베디드 시스템에 대한 관심증가에 따라, 데이터 중심의 응용프로그램과 특수목적에 부합하는 지능적인 시스템 개발의 필요성이 증대하고 있다. 최근 하드디스크를 플래시 메모리로 대체한 노트북 제품이 출시되고 있어 이를 뒷받침 한다.

플래시 메모리는 비휘발성이며 견고하고, 저 전력으로 동작이 가능하다. 접근 시간이 메인 메모리(RAM)와 유사할 만큼 빠르다. 따라서 대용량의 플래시 메모리는 지능형 임베디드 시스템에 적합하다.

플래시 메모리의 대용량화로, 파일시스템의 마운트 속도가 느려지고 있고, 예상치 못한 파워차단으로 인한 데이터 손실이 발생한다. 덮어쓰기가 불가능한 플래시 메모리는 쓰기전에 지움 작업을 하여야 한다. 이러한 플래시 메모리의 특성을 고려한 복잡한 파일 시스템의 개발이 진행되고 있다. 파일시스템은 쓰기작업 공간 확보시 많은 연산으로 시스템에 부담을 준다. 가장 무효한 블록을 지워 플래시 메모리 여유 공간을 확보하는 과정을 GC(Garbage Collection) 이라 한다[4,5,6]. 플래시 메모리의 일반적인 특성은 <표 1> 과 같다.

본 논문에서는 플래시 메모리의 데이터 공간 확보를 위한 GC 연산의 수행속도를 계산 한다. GC 연산 시간을 계산하기 위해, 쓰기, 읽기 연산 소요시간과 쓰기 공간 할당, 메인 메모리에 유지되는 파일 및 디렉터리 정보(Inodes), 블록(Block) 정보, 페이지 위치정보(Tnodes)의 연산 소요시간을 균등하게 부여하여 YAFFS, RFFS 파일시스템의 연산시간을 계산한다[3].

이 논문은 교육인적자원부 지방연구중심대학육성사업 (차세대물류IT기술연구사업단)의 지원에 의하여 연구되었음.

2장에서는 YAFFS 파일시스템의 GC 수행 절차에 대해 알아보고, 3장에서는 RFFS 파일 시스템과 RFFS 파일 시스템에서 지움연산 수행방법 을 설명한다. 4장에서는 RFFS의 GC 수행시 연산시간을 줄이는 방안을 제안한다. 5장에서는 결론을 제시한다.

<표 1> NAND 플래시 메모리의 특성(intel 28F640J3A)

특징	값
읽기 접근 시간	100~150ns
버퍼 이용 쓰기 시간 지움 블록 쓰기 시간	218㎲ /32bytes 0.8 sec/block
블록 지움 시간	1.0sec/block
최대 지움 횟수	100,000 times
지움 블록의 크기	128Kbyte
전력 소모량	대기 전력: 50~120㎂ 동작 전력: 15~70mA

2. YAFFS(Yet Another File System)

플래시 메모리는 느린 지움 속도 와 지움 횟수의 제한이라는 특성이 있다. 이러한 플래시 메모리의 특성을 고려한 다양한 플래시 파일시스템이 개발되었고, 플래시 파일 시스템은 빠른 마운트와 신뢰성 있는 데이터 유지는 파일 시스템의 중요한 요소가 되었다.

YAFFS 파일 시스템은 마운트 시 플래시 메모리 전체 스캔을 수행하여 메인 메모리에 정보를 유지한다. 파일 및 디렉터리의 정보를 가리키는 Inodes, 메모리에 저장되는 오브젝트 리스트, 파일내의 데이터 청크(chunk) 구조를 나타내는 Tnodes, 플래시 메모리에 저장되는 블록정보와 Tnodes 리스트, 앞선 정보의 포인트 정보를 나타내는 장치(Device) 정보로 구성된다.

YAFFS는 공간 확보를 위하여 가장 유효하지 않은 세그먼트를 지우는 그리디(Greedy) 지움 정책을 사용하며, 데이터의 신뢰성을 위한 저널링(Journaling)을 고려하고 있지 않다[4,7].

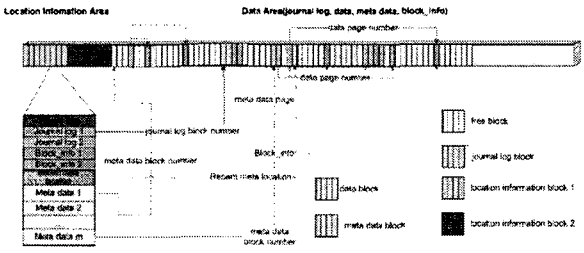
YAFFS의 GC 절차는 다음과 같다.

1.가장 유효하지 않은 세그먼트를 선택한다.
2.유효한 데이터 페이지(Page)를 새로운 세그먼트에 복사한다.
3.복사가 완료 되면, 블록을 지운다.
4.메모리의 Tnodes와 오브젝트(Object) 리스트를 갱신한다.

YAFFS의 GC 정책은 삭제시 새로운 블록에 복사하는 비용을 최소화 하지만, 특정 세그먼트에 집중해서 삭제가 일어 날 수 있기 때문에 수명을 단축시킬 수 있는 단점이 있다.

3. RFFS (Reliable Flash File System)

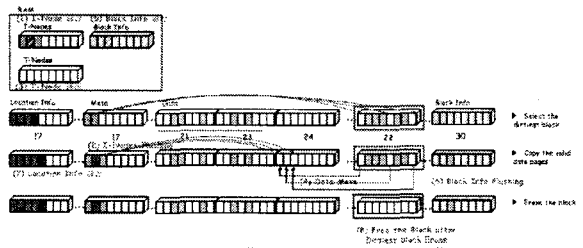
RFFS는 <그림 1>과 같이 NAND 전용 플래시 파일 시스템으로 빠른 마운트, 플래시 메모리의 수명을 고려한 사용 균등화(Wear-leveling)기법, 신뢰성 있는 데이터 유지를 위한 저널링(Journaling) 기법을 제공한다. 플래시 메모리에 파일 및 디렉터리의 상세정보를 나타내는 Inodes(Meta)정보, 각 블록의 상태정보를 나타내는 블록정보, 데이터 복구를 위한 로그 정보 및 앞선 각각 정보 포인터를 가리키는 위치(Location) 정보가 있다. 이 외에 메인 메모리에 유지되는 정보로 Inodes, 파일내 빠른 검색을 위한 Tnodes, 각 블록의 상태정보를 나타내는 블록 정보가 있다.



<그림 1> RFFS 파일 시스템 구조

메타(Meta), 로그, 블록 정보는 각각 4개의 블록들로 할당되고, 빠른 마운트 속도를 보장하기 위해 플래시 메모리의 1번째 블록에서 16번째 블록 까지 고정된 16개의 블록들이 예약된다[2].

YAFFS와 RFFS의 마운트 속도가 데이터 비율이 증가함에 따라 RFFS가 더 빠른 마운트 속도를 나타낸다. YAFFS의 플래시 메모리 전체 스캔을 통한 정보 수집 시간을 줄이기 위해, 각 블록의 상태 정보와 Inodes(Meta) 정보 포인터를 위치 정보 영역에 유지한다[2].



<그림 2> RFFS 데이터 영역의 GC 수행 구조

<그림 2>에서는 RFFS 파일 시스템의 데이터 블록의 GC 수행 방법을 보여준다. <그림 2>의 GC 수행절차를 요약하면 다음과 같다.

1. 메인 메모리와 플래시 메모리의 Inodes를 갱신
2. 선택된 블록의 유효한 페이지를 빈 블록으로 복사
3. 메인 메모리의 블록 정보 갱신
4. 메인 메모리와 플래시 메모리의 위치 정보 갱신
5. 선택된 블록 지움
6. 플래시 메모리의 블록 정보 갱신

GC 수행 연산은 메모리 접근시간, 페이지단위 쓰는 시간, 블록 지움 시간을 고려하면, 다음과 같은 계산식이 구해진다.

(식 1)은 YAFFS GC 수행 연산식을 나타내고, (식 2)는 RFFS GC 수행 연산식을 나타낸다.

```
YAFFS_GC =  RAM_ACCESS_TIME * YAFFS_RAM_ACCESS_COUNT +
      (VALID_PAGE_COUNT + 1) * WRITE_PAGE_TIME +
      BLOCK_ERASE_TIME
YAFFS_GC =  inCase_data_Block_alloc * YAFFS_GC +
      YAFFS_GC
                                                  (식 1)
```

```
RFFS_GC =  RAM_ACCESS_TIME * RFFS_RAM_ACCESS_COUNT +
      (i + RFFS_BLOCKINFO_WRITE_COUNT_PER_ERASE +
      RFFS_LOCATIONINFO_WRITE_COUNT_PER_ERASE +
      RFFS_META_WRITE_COUNT_PER_ERASE) *
      WRITE_PAGE_TIME + BLOCK_ERASE_TIME
RFFS_GC = inCase_Reserve_alloc * (RAM_ACCESS_TIME *
      2 + RFFS_BLOCKINFO_WRITE_COUNT_PER_ERASE +
      VALID_COUNT_PER_BLOCK * WRITE_PAGE_TIME +
      BLOCK_ERASE_TIME) * META_ALLOC_COUNT +
      RFFS_GC
                                                  (식 2)
```

RFFS는 YAFFS에 비해 지움 연산의 과정이 메타, 블록, 데이터, 로그, 위치정보 갱신 등 4단계를 더 거쳐서 이루어 지는 경우가 발생한다. 따라서, RFFS의 GC 수행은 YAFFS에 비해 최대 5배의 연산 시간의 차이가 난다.

4. RFFS 지움연산 최소화 방법

RFFS의 연산 과정이 다른 파일 시스템에 비해 더 많은 단계를 거치므로 GC 수행시 더 많은 연산시간이 필요하다.

기존의 파일 시스템은 GC 수행을 블록 단위로 수행하고 있고, 약 1초의 지움 연산 시간이 소요된다.

NAND 플래시 메모리는 셀이 직렬로 연결되어 있어, 읽기 속도에 비해 쓰기 속도가 상대적으로 느리다. 그리고, 연속된 블록들을 한번에 지움 연산을 수행하는 경우와 하나의 블록을 지움연산 수행하는 경우의 연산 시간이 유사하다. 이러한 특성을 고려하여, 플래시 메모리의 모델에 따라 지움 시간, 읽기 접근 시간의 차이가 있으나, 연산시간 계산을 위하여 <표 1>을 참고로 페이지 쓰기(page write) 시간은 3.5ms, 블록

지움 시간은 800ms로 가정하였다. 메모리 접근 시간은 무시할 정도의 아주 짧은 시간으로 고려하지 않았다. GC 연산 시간을 계산한 결과 다음과 같은 결과를 얻을 수 있다.

<표 2> 1개 Block단위 지움 연산시간

(ms)

	최소연산시간	최대연산시간
YAFFS(1개 블록)	810	1620
RFFS(1개 블록)	817	7264
RFFS(4개 블록)	817	3837

<표 2>에서 보는바와 같이, RFFS는 1개 블록 단위로 지움 연산을 하는 경우와 4개의 연속된 블록 단위로 지움 연산을 계산한 경우, 최대연산시간이 각각 7264ms, 3837ms로 약 3초가 개선되었다. 그리고, YAFFS와 RFFS의 최대연산시간을 비교하면 각각 1620ms, 3837ms로 RFFS가 YAFFS에 비해 2217ms 만큼 연산시간이 더 소요된다. 그러나, RFFS는 메타, 블록, 로그 정보를 한번에 4개의 블록을 할당 하고, YAFFS는 한번에 1개의 블록을 할당한다. 따라서, RFFS의 GC 연산횟수는 YAFFS에 비해 적게 발생한다. GC 연산이 많이 발생 할수록 GC 연산 총시간 차이가 줄어든다.

그리고, RFFS의 GC 수행시 외부 단편화로 인한 연산지연을 예방하고, 4개의 블록 할당 공간 확보 시 GC 수행에 의한 부하를 줄이기 위해서, 무효화 주기가 짧은 데이터를 인접한 영역에 할당되도록 하기 위해서 가중치를 부여하고, 높은 가중치를 가진 데이터는 예약영역을 확보한다. RFFS 파일 시스템에서 볼 때, 데이터의 종류는 메타정보, 데이터, 로그정보, 블록정보, 위치정보로 구분할 수 있고, 앞선 데이터의 종류를 무효화 주기가 짧은 순으로 정렬하면 블록정보, 로그정보, 메타정보, 데이터, 위치정보 순이다. 데이터의 갱신은 메타정보, 로그정보, 블록정보, 위치정보의 갱신을 요구하고, 메타의 갱신은 로그정보, 블록정보, 위치정보의 갱신을 요구한다. 로그 정보는 블록정보, 위치정보의 갱신을 요구한다. 마지막으로, 블록정보는 위치 정보의 갱신을 요구한다. 이러한 데이터의 특성을 고려해 볼 때, 블록, 로그 정보는 무효화 되는 주기가 짧은 데이터(Hot 데이터)의 특성을 가지고, 메타정보와 데이터는 블록 정보와 로그 정보에 비해 무효화 주기가 긴 데이터(Cold Data) 특성을 가진다. 위치 영역의 경우 자체적으로 순차적인 갱신과 지움 연산이 발생함으로 고려하지 않는다. 메타정보를 로그정보와 블록정보 영역의 할당 공간을 데이터와 메타정보의 일정한 할당주기에 따라 예약하고 할당한다면 로그와 블록 정보의 빠른 무효화로 쓰기 공간 확보가 용이하고, GC 연산 시간이 줄어든다. 또한, 데이터 쓰기 공간이 충분한 경우에 파일 연산시 블록내 모든 페이지가 무효한 경우만 가비지 컬렉션(Garbage Collection)을 수행하여 데이터 공간을 확보하고, 나머지 경우는 GC 수행을 위한 임계값(Threshold) 만큼 블록이 존재 하지 않을 경우에 수행 한다.

5. 결론

본 논문에서는 빠른 마운팅과 플래시의 수명, 데이터의 신뢰성을 보장해 주기 위해 개발된 RFFS 파일 시스템에서 GC 수행의 연산에 대한 평가를 하였다.

RFFS는 YAFFS에 비해 GC 연산 최대시간이 2배이상 소요되지만, RFFS가 할당하는 블록의 수가 4개여서 한번에 하나의 블록만 할당하는 YAFFS에 비해 누적되는 GC 연산횟수가 적다는 것을 알았다. RFFS의 GC 수행 연산으로 인한 부하를 줄이기 위해서, 임계값 이전에는 무효화 데이터만 존재하는 블록만 GC를 수행하고, 임계값에 도달한 경우에 GC를 수행한다. 또한, 무효화 주기가 짧은 로그정보, 블록정보의 특성을 고려하여 무효화 주기가 긴 데이터 할당시 무효화 주기가 짧은 데이터 블록을 위해 예약하는 방법을 제안했다.

향후에는 제안한 GC 수행 연산을 고려한 기법을 실제 파일시스템에 구현하고, 실험을 통한 GC의 성능 평가를 할 예정이다. 그리고, GC 수행시 예기치 못한 전원차단으로 인한 데이터 오류가 발생할 경우 처리할 수 있는 기법을 연구 할 것이다.

참고 문헌

[1] 김한준, 이상구 "신뢰성있는 플래시메모리 저장시스템 구축을위한 플래시메모리 저장 공간 관리방법" 정보과학회논문지: 시스템 및 이론 제27권 제6호(2000.6)

[2] 이태훈, 박송화, 김태훈, 이상기, 이주경, 정기동 " 임베디드 시스템을 위한 신뢰성 있는 NAND 플래시 파일 시스템의 설계", 정보처리학회논문지 제12-A권 제7호(2005. 12)

[3] Li-Fu Chou, Pangfeng Liu, "Efficient Allocation Algorithms for FLASH File Systems", Proc. of IEEE Conference. Parallel and Distributed Systems. (2005)

[4] M. L. Chang, Paul C. H. Lee, Ruei-Chuan Chang, "Cleaning Policies in Mobile Computers Using Flash Memory", Accepted by Journal of System and Software.

[5] M. L. Chiang, C. H. Paul, and R.C Chang. "Mange flash memory in personal commuicate devices", Proc. of IEEE Symp. (1997)

[6] A. Kawaguchi, S. Nishioka, H. Motoda, " A flash memory based file system", Proc. of Computer Software and Applications Conferece.(1999)

[7] David Woodhouse, "The Journalling Flash File System", http://sourceware.org/jffs2/jffs2.pdf

[8] Wookey, "YAFFS A NAND-flash filesystem", http://www.ukuug.org/events/linux2004/programme/paper-Wookey-1/yaffs2004talk.pdf