JFFS2 Garbage Collector의 설계

김기영° 정성욱 손성훈 신동하
상명대학교 일반대학원 컴퓨터과학과
{ kgykingdom°, director, shson, dshin }@smu.ac.kr

Design of Garbage Collector for JFFS2

Ki-Young Kim°, Sung-Wook Jung, Sung-Hoon Sohn, Dong-Ha Shin
Department of Computer Science, Graduate School, Sangmyung University

요 약

플래시 메모리는 PDA 등과 같은 모바일 장치에 많이 사용되는 저장 매체이다. 이 매체는 덮어 쓰기가 불가능하다는 특징으로 인해 흔히 JFFS2와 같은 전용 파일 시스템을 사용하게 된다. 현재의 JFFS2 파일 시스템은 Garbage Collection 시 많은 자원과 시간을 소비하는 문제점을 가지고 있다. 본 논문에서는 JFFS2 Garbage Collector에서 유효한 노드에 대한 다시 쓰기 작업을 개선한 새로운 Garbage Collector를 설계하였고, 이에 대한 성능 평가를 수행하였다.

1. 서 론

플래시 메모리는 여러 프로그램이 동시에 읽기와 쓰기를 할 수 있는 EEPROM 형태의 저장 매체이다. 또한, 전원이 없이도 데이터를 유지할 수 있고 다시 쓰기가 가능한 메모리 칩이다. 이러한 플래시 메모리는 작은 크기와 낮은 전력 소비, 충격에도 잘 견디는 특징 등으로 인해 배터리를 사용하는 다양한 모바일 장치에 널리 쓰이고 있다.

그러나, 플래시 메모리는 이러한 특징에도 불구하고 쓰기 속도와 지우기 속도가 느리다는 치명적인 약점을 가지고 있다. 특히 지우기 연산의 경우 그 단위가 읽기나 쓰기 단위와 틀리기 때문에, 지우기 연산을 하는 과정에서 다른 곳으로 옮겨 쓰지 않아도 되는 유효한 데이터를 이동시키는 작업을 해야 한다. 이러한 현상은 플래시 메모리가 일반 디스크 등과는 달리 매체의 특정 페이지에 수정된 데이터를 덮어쓰는 것이 불가능 하다는 점 때문에 발생하게 된다. 즉 기존에 저장된 내용을 수정하려면 해당 데이터를 삭제 한 후 플래시의 빈 공간에 다시 쓰는 작업을 해야 한다.

리눅스 운영체제는 JFFS2 라는 플래시 메모리 전용 파일 시스템을 제공한다. JFFS2 파일 시스템은 앞서 언급한 플래시 메모리의 특성에 맞는 Log Structured 파일 시스템이다 [4]. JFFS2에서는 노드라는 단위로 읽기와 쓰기가 이루어지며, Obsolete 상태의 노드를 삭제 하기 위해서 Garbage Collector를 사용하고 있다. JFFS2의 각 Eraseblock의 한 노드가 수정에 의해 삭제 되면 JFFS2 는 노드를 Obsolete 시키게 된다. 이렇게 Obsolete된 노드의 횟수에 따라서 이 노드를 포함하는 Eraseblock은 여러 개의 dirty list 중 하나에 매달리게 된다. 이후 매체의 빈 공간이 임계치보다 떨어지면 JFFS2의 Garbage Collector가 동작하게 되고, Garbage Collector는 적절한 dirty list를 선택, 빈 공간을 늘이는 작업을 수행하게 된다.

그러나 이런 Garbage Collector에 의한 노드의 삭제는 상당히 많은 시간과 비용이 드는 작업이기 때문에 Garbage Collector가 불려지는 횟수를 최소화하는 것이 전체 성능상 매우 중요하다. 본 논문에서는 JFFS2 Garbage Collection시 유효한한 노드의 다시 쓰기 작업에서 오는 오버헤드를 최대한 줄이기 위한 기법을 제안한다.

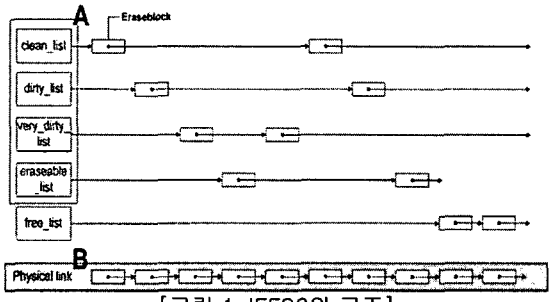
2. 관련 연구

JFFS2의 단순한 Garbage Collection 방법에서 오는 문제점은 유효한 데이터를 반복적으로 다른 장소에 복사하게 되는 점이다. 또한, 이로 인해 플래시 메모리 페이지의 사용 횟수를 불필요하게 높이게 된다. 이런 문제점을 막기 위해 시도된 방법으로는 [2]에서 제안된 고빈도(Hot)의 데이터와 저빈도(Cold)의 데이터를 분류하는 기법을 들 수 있다. 여기서 고빈도(Hot) 데이터는 수정이 자주 일어나는 데이터로 파일에 대한 메타데이터 같은 것을 의미한다. 반면에 저빈도(Cold) 데이터는 프로그램의 코드와 같이 갱신이 거의 일어나지 않는 데이터를 의미한다. 또한, [3]에서는 고빈도(Hot)의 데이터와 저빈도(Cold)의 데이터를 서로 다른 곳에 저장하는 기법을 다루고 있다. 이 기법은 서로 같은 다른 빈도의 데이터들이 모여 있을 경우 데이터가 계속해서 복

사되어 움직이는 것을 방지하고 있다.

3. JFFS2 Garbage Collector의 개선

3.1. JFFS2의 Garbage Collector



[그림 1 JFFS2의 구조]

JFFS2의 Garbage Collection 구조는 그림 1과 같다. 그림 1에서 A는 JFFS2의 메모리상의 리스트들을 나타낸다. 실제 JFFS2의 Eraseblock들은 그림 1 B처럼 다음번 물리적 Eraseblock를 가리키는 방식으로 구성되어 있다. 최초의 리스트들은 마운트 시 매체 전체를 scan함으로써 구성된다. Clean_list는 모든 노드가 유효한일 경우, dirty_list와 very_dirty_list는 Eraseblock의 Obsolete된 노드의 횟수에 따라서 매달리게 되고, eraseable_list는 바로 erase 작업을 수행해도 되는 Eraseblock들이 매달리게 된다. Free_list는 erase작업이 끝난 사용 가능한 Eraseblock들이 매달리게 된다. Garbage Collector는 그림 1 A의 4개의 list(clean_list, dirty_list, very_dirty_list, eraseable_list)중에서 jiffies 값을 128로 나눈 값에 따라 대상을 랜덤하게 선택하게 된다. Garbage Collection이 끝난 Eraseblock은 free_list에 매달리게 되고 write는 free_list에서 새로운 Eraseblock을 할당해서 write를 수행 하게 된다.

이런 방법은 구현이 매우 간단한 장점을 가지고 있지만, 저장된 데이터의 특성을 전혀 고려하지 않아 한번의 Garbage Collection에 상당히 많은 자원과 시간을 소비 하게 된다. 특히 한 Eraseblock에 유효한 데이터와 Obsolete된 데이터가 함께 있는 경우 이 Eraseblock은 Garbage Collection의 대상이 된다. 이때 유효한 데이터는 새로운 Eraseblock을 할당 받아 복사하게 되는데, 이 유효한 데이터가 프로그램 코드와 같이 변하지 않는 부분이라고 할 지라도 (그래서 항상 유효한 데이터라고 할 지라도) 반복적으로 새로운 Eraseblock에 복사하게 된다. 이런 작업은 JFFS2 Garbage Collection이 과도한 쓰기 작업을 하게 되는 가장 큰 이유가 된다.

3.2 새로운 JFFS2 Garbage Collector

3.1절에서 언급한 문제점을 개선하기 위해서 본 논문에서는 [3] 의 고빈도의 데이터와 저빈도의 데이터를 서로 다른 장소에 쓰는 것에 착안하여 단순한 Garbage Collection 에서 오는 과도한 데이터 쓰기를 줄이고 플래시 각 Eraseblock의 lifetime을 늘리는 방법을 고안하였다.

개선된 알고리즘에서는 Garbage Collector가 오랫동안 유효한 데이터 노드를 새로운 Eraseblock으로 복사하는 경우, 이 유효한 데이터 노드를 현재 Free 상태인 Eraseblock 중에서 사용 횟수가 가장 높은 곳에 쓰게 한다. 이렇게 사용 횟수가 높은 Eraseblock에 쓰기 위해 Garbage Collector에 사용 횟수가 높은 Eraseblock에 쓰기 위한 버퍼를 하나 만들고, 일정 횟수 이상의 다시 쓰기가 이루어진 데이터 노드는 이 버퍼에 넣게 된다. 그리고, 버퍼가 가득 차게 되면, 새로운 Eraseblock을 할당하여 버퍼의 내용을 넣고, 새로 만든 gc_copy_list에 매달게 된다. 여기서 gc_copy_list는 clean_list보다 더 적은 확률로 Garbage Collection이 수행된다.

또한, 다시 쓰기를 하려는 데이터 노드가 Garbage Collection을 통해 다른 Eraseblock으로 몇 번 다시 쓰기가 되었는지는 각 노드의 메타데이터에 기록한다. 이 횟수는 데이터 노드를 사용 횟수가 높은 Eraseblock에 쓰기 위한 버퍼에 넣을 것인지 말 것인지를 결정하게 되는 값이 된다. 이렇게 사용률이 높은 Eraseblock에 오랫동안 유효한 데이터를 저장하게 되면, 이 데이터는 수정될 확률이 적으므로 비교적 오랫동안 Garbage Collection이 될 가능성이 적어지게 된다. 또한 이러한 과정에서 사용 빈도가 높은 Eraseblock을 잠시 동안 사용을 안하게 됨으로써 전체적인 사용률 평준화를 이루게 된다. 또한, 계속해서 유효한 데이터 노드를 향후 유효할 가능성이 큰 데이터들과 함께 사용률이 높은 Eraseblock에 따로 저장 함으로써 Garbage Collector가 다시 쓰기를 위해 사용하는 시간과 자원을 절약 할 수 있게 된다.

4. 성능 평가

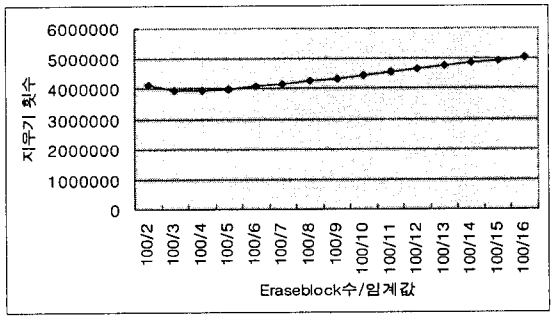
본 논문에서 제안된 기법의 목적은 Garbage Collector가 유효한 노드를 새로운 Eraseblock에 다시 쓰는 작업을 최대한 줄이는 것이다. 제안된 기법의 성능을 확인하기 위해 Redhat Linux 9.0 상에서 최신 커널 버전인 linux-2.6.11.8에 포함된 JFFS2를 기초로 시뮬레이터를 구현하여 제안된 기법에 대한 성능 평가를 수행하였다.

이 성능 평가를 위해 읽기와 쓰기, 지우기 10,000,000개 작업에 1:2:1의 비율로 구성된 워크로드를 사용하였고, 각각 512KB의 Eraseblock 50개(25.6 M)와 100개(51.2 M)로 이루어진 플래시 메모리를 가정하여 JFFS2의 Garbage Collector와 제안된 Garbage Collector를 비교하였다.

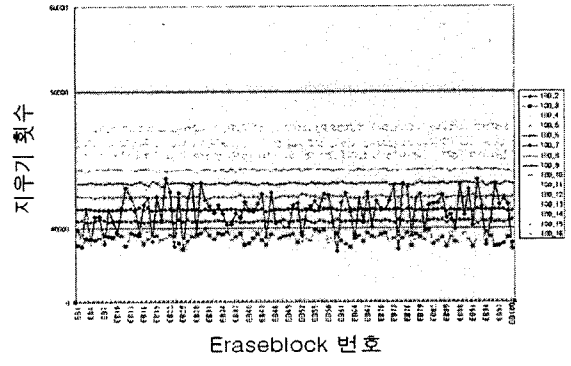
특히, Eraseblock이 100개인 플래시 메모리의 경우 저장 공간이 더 넓기 때문에 자주 변경 되지 않는 data가 더 오랫동안 머물고 Garbage Collector에 의한 다

시 쓰기 작업도 많다. 따라서 데이터 비교의 경우 주로 Eraseblock 이 100개인 경우를 사용했다. 또한, 유효한 노드의 다시 쓰기 작업의 횟수는 전체 지우기 횟수를 줄이는 것과 동일하다고 보고, 각 Garbage Collector의 지우기 횟수를 비교하였다. 그리고, 각 블록의 평준화 정도 비교를 위해 각 블록에 대한 지우기 횟수 또한 비교하였다.

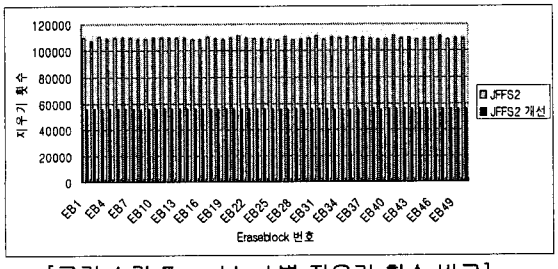
개선된 JFFS2 Garbage Collector는 다시 쓰기를 줄이기 위해 유효한 노드의 다시 쓰기가 이루어 질 경우 각 Eraseblock의 다시 쓰기 카운터를 1만큼 증가시킨다. 이 경우 노드의 다시 쓰기 카운터가 얼마 이상 되어야 현재 free_list에 들어 있는 Eraseblock중에 사용률이 가장 높은 Eraseblock에 다시 쓰기를 할 것인지에 대한 값을 결정해야 한다.



[그림 2 Eraseblock의 다시쓰기 횟수 설정 값에 따른 전체 지우기 횟수]



[그림 3 각 Eraseblock의 다시쓰기 횟수 설정 별 지우기 횟수]



[그림 4 각 Eraseblock별 지우기 횟수 비교]

그림 2와 그림 3은 이러한 값을 찾기 위한 실험으로, 100개의 Eraseblock으로 4개의 다시 쓰기 값을 임계치로 두었을 때가 3,921,265번으로 지우기 횟수가 가장 작기 때문에 다시 쓰기 횟수 4회를 임계치로 설정되었다. 또한, 비슷한 지우기 횟수를 보여주는 3개의 경우는 그림 3에서 값이 불안정 한 변화를 보여 보다 안정적인 값인 4를 선택하게 되었다.

그림 4에서 동일한 연산을 100개의 Eraseblock에 주었을 때, 원래 JFFS2보다 개선된 JFFS2에서 지우기 연산의 횟수가 2배 이상 줄어든 것을 관찰 할 수 있다. 또한, 각 Eraseblock의 지우기 횟수의 평준화 또한 개선된 Garbage Collector의 경우가 뛰어남을 관찰할 수 있다.

5. 결 론

본 논문에서는 JFFS2의 Garbage Collector를 수정하여 보다 효과적인 Garbage Collector를 설계하였고, 이에 대한 성능 평가를 수행하였다.

새로 제안된 Garbage Collector는 변경이 없는 데이터 노드와 변경이 자주 일어나는 데이터 노드를 구분하고, Garbage Collection시에 변경이 없는 데이터 노드를 사용률이 높은 Eraseblock에 저장함으로써 플래시 메모리의 사용률 평준화를 꾀하고 있다. 또한 변경이 없는 데이터를 Garbage Collection 하지 않게 됨에 따라, Garbage Collector의 동작으로 인해 시간과 자원이 낭비되는 것을 방지하고 있다.

하지만, 오랫동안 유효한인 데이터 노드가 사용률이 높은 Eraseblock에서 계속해서 수정이 없이 남아있게 될 경우, 이 Eraseblock의 사용률이 다른 Eraseblock의 사용률 보다 낮아지면서 역효과가 발생할 가능성이 남아있으며, 현재 이 문제를 해결하기 위한 연구가 진행 중이다.

6. 참고 문헌

[1] 정하용, 김진수, 한환수, 최기선, " JFFS2를 위한 효율적인 Garbage Collector의 설계" , 한국정보과학회, 2004.

[2] M. L. Chiang, Paul C. H. Lee, and R. C. Chang, " Data Management in a Flash Memory Based Storage Server " , International Computer Symposium(ICS' 98), 1998.

[3] Atsuo Kawaguchi, Shingo Nishioka, and Hiroshi Motoda, " A Flash-Memory Based File System" , Usenix Technical Conference, 1995.

[4] Mendel Rosenblum and John K. Ousterhout, " The Design and Implementation of a Log-Structured File System" , ACM Transactions on Computer Systems 10(1), 1992