낸드 플래시 파일 시스템의 성능 향상을 위한 MFU 페이지 할당 기법

유태경° 김기창
인하대학교 정보통신공학부
r4bbit@inhain.net, kchang@inha.ac.kr

MFU Page Allocation Technique To Improve Performance for NAND Flash File System

Taikyung Yu° Kichang Kim
School of Information and Communication Engineering, Inha University

낸드 플래시 메모리는 소형이면서 외부 충격에 강하여 디지털 기기에 적합하며, 특히 전력 소비가 적어 전원 공급에 제약이 있는 모바일 기기에 적합하다. 그러나 기존의 저장매체와 구별되는 다음과 같은 제한 사항들을 가진다. 첫째, 덮어 쓰기를 위해서는 삭제 연산이 선행 되어야 하고 둘째, 삭제 연산과 읽기, 쓰기 연산의 단위가 다르다. 셋째, 삭제 연산이 다른 연산에 비해 느리고 전력 소비가 크며 넷째, 블록마다 삭제 연산의 횟수가 제한된다[1]. 그것을 극복하기 위해 LFS(Log-structured File System)에서는 덮어 쓰기 연산을 추가 쓰기 연산으로 변형하여 처리하는 외부 갱신 기법을 사용한다[2]. 그러나 그 과정에서 생긴 무효화 데이터들을 회수하여 다시 쓰기 가능한 상태로 전환하는 블록 클리닝이 요구되고, 그 블록 클리닝 비용 및 횟수를 줄이는 것이 파일 시스템의 성능에 큰 영향을 미친다[3, 4, 5].

블록 클리닝 성능을 높이기 위한 기존 연구들의 기본적인 아이디어는 데이터 중에 Hot 데이터(최근에 자주 업데이트 되는 데이터)를 판단하여 같은 블록(Hot 블록)에 할당하는 것이다[3, 4]. Hot 데이터는 무효 데이터가 될 가능성이 크기 때문에 Hot 블록의 대부분은 무효 데이터일 가능성이 크다. 그러므로 Hot 블록을 클리닝하면, 많은 무효 데이터들을 회수하여 자유 공간을 확보함으로써 결과적으로 블록 클리닝 횟수를 줄일 수 있고, 또한 유효 페이지가 적기 때문에 블록 클리닝 시에 드는 재복사 비용을 줄일 수 있다. 그러므로 논의의 중심은 효과적으로 Hot 데이터를 판단하여 Hot 블록에 위치시키는 기법에 있고 본 논문에서는 페이지 단위로 Hot 데이터를 판단하는 MFU(Most Frequently Updated) 알고리즘을 제안한다.

MFU는 파일의 논리 페이지에 대한 정보를 윈도우 리스트에 기록하고, 페이지 할당 요청 시에 그것을 기반으로 해당 논리 페이지가 Hot 페이지인지를 판단하는 알고리즘이다. 윈도우는 논리 페이지의 업데이트 빈도를 관리하기 위해 link(윈도우 리스트를 만들기 위한 링크), chunkInInode(업데이트 되는 논리 페이지 번호), update_cnt(업데이트 횟수)로 구성된다. 파일 업데이트 시에 업데이트 되는 논리 페이지가 이미 윈도우 리스트에 존재하면 update_cnt를 증가시키고 윈도우를 최신의 위치로 이동(window_head 쪽)시키며, 그렇지 않으면 새로운 윈도우를 할당하고 윈도우 리스트에 연결한다. 그리고 윈도우 리스트는 미리 정의된 최대 개수(MAX_WINDOW_SIZE)이하의 윈도우로 구성되며, 파일 당 하나 존재한다. 그러므로 윈도우의 개수(윈도우 사이즈)가 최대일 경우 업데이트 횟수가 가장 적고 오래된 윈도우를 희생 윈도우로 선택하여 삭제한 후에 새로운 윈도우를 추가한다. 또한 윈도우 리스트의 전체 업데이트 횟수가 미리 정의된 최대 횟수(MAX_TOTAL_UPDATE_CNT)이상이 되면 윈도우 리스트를 초기화시켜 최신 정보를 유지한다. 이렇게 구성된 윈도우 리스트를 기반으로 페이지를 할당할 때, 해당 논리 페이지의 업데이트 횟수가 윈도우 리스트의 평균 업데이트 횟수보다 크거나 같은 경우 Hot 페이지로 판단하고 이를 Hot 블록에서 할당한다.

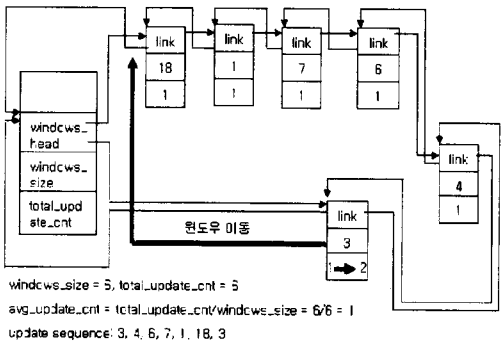

그림 1. 윈도우 리스트 1

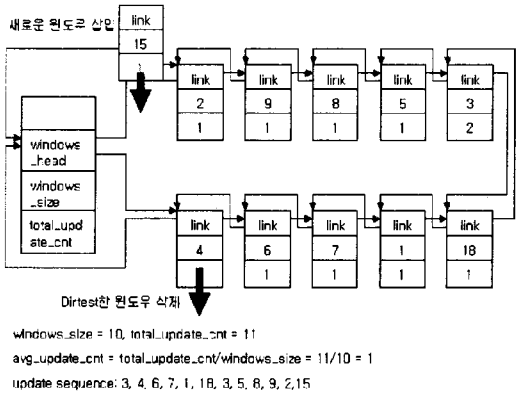

그림 2. 윈도우 리스트 2

그림 1은 해당 파일의 논리 페이지가 3, 4, 6, 7, 1, 18, 3'의 순서로 업데이트 되면서 만들어진 윈도우 리스트이다. 논리 페이지 3'이 업데이트 되는 시점에서, 윈도우 리스트의 평균 업데이트 횟수는 6/6=1이고, 논리 페이지 3의 윈도우의 업데이트 횟수는 1이므로, 즉 해당 윈도우의 업데이트 횟수가 윈도우 리스트의 평균 업데이트 횟수보다 크거나 같으므로 페이지를 Hot 블록에서 할당 한다. 그리고 해당 윈도우의 업데이트 횟수를 증가시키고, 최신의 위치로 이동시킨다. 그 다음으로 논리 페이지가 5, 8, 9, 2, 15의 순서로 업데이트 되면(그림 2), 논리 페이지 2가 업데이트 되는 시점에서 윈도우 리스트의 크기는 MAX_WINDOW_SIZE가 되므로 논리 페이지 15가 업데이트 되는 시점에서 업데이트 횟수가 가장 작고, 그 중에 가장 오래 전에 업데이트된 논리 페이지의 정보가 기록된 윈도우를 해제한다. 그리고 그것의 업데이트 횟수를 윈도우 리스트의 전체 업데이트 횟수에서 뺀 후, 논리 페이지 15를 위한 새로운 윈도우를 할당하고, 윈도우 리스트에 포함시킨다.

실험을 위해 MFU 페이지 할당기법을 YAFFS(Yet Another Flash File System)에 구현 하였고, 이를 임베디드 리눅스에 포팅하였다. 그리고 워크로드는 VFS(Virtual File System)를 통해 open(), write(), read(), close()등의 사용자 인터페이스를 가지고 애플리케이션으로 구현하였다. 실험 결과, 업데이트 접근 패턴의 지역성이 거의 없는 Uniform Pattern인 경우, 업데이트 횟수가 100,000일때 블록 삭제 횟수가 약 1.4%가 감소하였고, 블록 삭제 횟수당 유효 페이지 복사 횟수는 약 11.6%가 감소하였다. 그리고 Hot-and-Cold Pattern의 경우, 업데이트 횟수가 150,000일때 블록 삭제 횟수는 약 11%가 감소하였고, 블록 삭제 횟수당 유효 페이지 복사 횟수는 약 20.1% 감소하였다. 이 결과로 보아 MFU 알고리즘에 의해 데이터를 효과적으로 재배치함으로써 기존 기법보다 비용과 실행 횟수의 측면에서 성능이 향상되고, 파일들의 업데이트 접근 패턴이 지역성을 가질수록 더 크게 향상 되는 것을 알 수 있다.

참고 문헌

[1] Samsung Electronics, "64M x 8 Bit NAND Flash Memory".

[2] K. Kawaguchi, S. Nishioka, and H. Motoda, "A Flash-memory based File system", USENIX, pp.155-164, 1995.

[3] M-L. Chang, P. C. H. Lee and R-C. Chang, "Using data clustering to improve cleaning performance for flash memory", Software, Vol.29, pp.267-290, 1999.

[4] J. Wang. Y. Hu, "WOLF-A Novel Reordering Write Buffer to Boost the Performance of Log-Structured File Systems", USENIX, pp.145-158, 2004.

[5] L. P. Chang, T. W. Kuo and S. W. Lo, "Real-time Garbage Collection for Flash Memory Storage Systems of Real time Embedded Systems", ACM, Vol.3, pp.837-863, 2004.