커널 레벨 가변 길이 블록 파일 시스템

유영준*, 고영웅*
*한림대학교 컴퓨터공학과
e-mail:willow72@hallym.ac.kr

Design and Implementation of Kernel-Level File System Using Variable-Length Blocks

Young-Jun Yoo*, Young-Woong Ko*
*Dept of Computer Science, Hallym University

요 약

일반적인 에디터 프로그램과 운영체제를 사용하여 데이터를 편집할 경우, 일부 수정에도 모든 데이터를 다시 저장하고 있다. 본 논문에서는 이러한 기존 파일시스템의 문제점을 개선하기 위해 가변 길이 블록의 개념을 커널 레벨에 적용하여 효율적으로 수정된 데이터를 다시 쓸 수 있도록 하였다. 가변 블록은 데이터 삭제가 발생 했을 경우 나머지 블록은 그대로 유지하고 수정된 블록만 다시 저장하며, 삭제된 데이터로 인해 생기는 빈 공간의 그대로 유지하는 방법을 말한다. 이 개념을 기존 리눅스에서 사용하는 ext4파일 시스템에 적용하여 시스템을 구축, 실험하였으며 결과적으로 쓰기 연산을 비롯해 CPU사용량에서 크게 성능을 향상시켰다.

1. 서론

운영체제에서 파일시스템[1]은 데이터를 효율적으로 저장, 관리하기 위한 필수적인 요소이다. 데이터들은 하드 디스크에 서로 인접하거나 멀리 떨어진 여러 블록에 나뉘어 저장되며, 유저 어플리케이션에서 처리 될 때는 효율성을 위해 하나의 연속적인 바이트 스트림(byte stream)으로 처리된다. 하지만 이러한 처리방식은 수정된 데이터를 다시 저장하는 과정에서 비효율적인 부분이 존재할 수 있다. 예를 들어 *VI*에디터와 같은 에디터 프로그램에서 데이터의 일부를 삭제하고 다시 저장할 경우 스트림 상에서는 모든 데이터가 삭제된 만큼 앞으로 당겨지는 변화가 발생한다[2]. 그리고 파일시스템에서는 이러한 수정을 고려하지 않고 모든 데이터를 다시 쓰는 방법을 사용하고 있다.

따라서 본 논문에서는 가변 길이 블록의 개념[3]을 커널 레벨에 적용하여, 수정된 데이터에 대해 다시 쓰기 연산을 수행할 때 쓰기 연산량을 최소화시키고자 한다.

본 논문의 구성은 다음과 같다. 제일 먼저 2장 관련연구에서는 기존 파일 시스템의 문제점과 유저 레벨 파일 시스템에 적용된 가변 길이 블록에 대해 설명한다. 그리고 3장 시스템 구현에서는 가변 블록을 적용하기 위한 구조도와 알고리즘을 설명하며, 4장 실험 및 결과에서는 가변 블록을 적용한 ext4 파일 시스템과 일반 ext4 파일시스템의 성능을 비교하여 효율성을 증명한다. 마지막으로 5장 결론 및 향후 연구로 맺는다.

1) 이 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 2014R1A2A1A11054160). 또한 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.2012R1A1A2044694)

2. 관련 연구

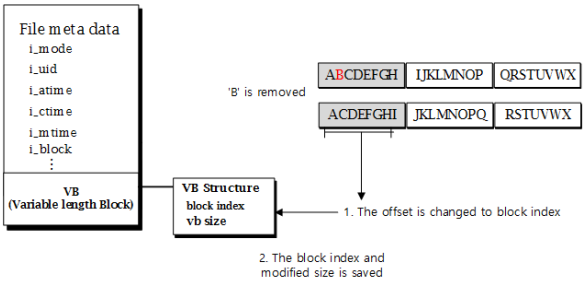
기존의 파일 시스템과 일반적으로 사용되는 *VI*에디터에서 일부의 데이터를 수정하고 다시 저장할 경우 파일은 모든 내용이 다시 써지며 많은 쓰기 연산양을 발생시킨다. 이러한 문제점을 해결하기 위해 유저 레벨 가변 길이 블록 파일 시스템에서는 삭제되는 데이터만큼 범퍼 영역을 생성해 이후 데이터들이 밀리는 현상을 막는다. 유저가 특정 부분(offset)의 데이터를 수정하고 다시 저장할 경우 시스템은 이 수정된 이 위치의 정보를 블록의 인덱스로 변경해 파일 시스템으로 전달한다. 그리고 파일을 저장하는 단계에서 수정이 발생한 블록만 다시 저장하고 삭제된 데이터로 인해 당겨진 데이터만큼 블록의 뒷부분은 범퍼 영역을 생성한다. 반면 나머지 블록들은 그대로 유지함으로써 적은 쓰기 양으로 파일을 수정할 수 있다.

3. 시스템 구현

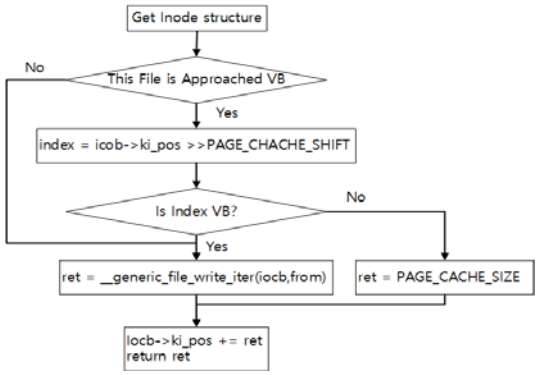
본 논문에서는 커널 내의 가변 길이 블록 개념을 증명하기 위해 리눅스 커널 4.4.1 버전에서 ext4 파일시스템을 수정하였으며, 가변 길이 블록의 정보를 커널로 전달하기 위해 별도의 *vlfs_modify()* 시스템 콜을 추가하였다.

그림1는 가변 길이 블록을 적용하기 위해 수정된 ext4의 *Inode*와 새로 추가된 시스템의 순서를 보이고 있다. 수정된 파일의 위치와 길이를 저장하기 위해 *VB*자료구조가 추가되었으며, 시스템 콜이 발생하면 유저 어플리케이션의 데이터 스트림 중 수정된 오프셋을 블록의 인덱스로 변환

하고 수정된 길이와 함께 저장된다. 현재 시스템은 프로토타입으로써 해당 시스템 콜에 수정된 오프셋과 길이는 매개변수를 통해 임의로 전달한다.



(그림 1) 새로 추가된 *VB*자료구조와 수정된 데이터의 저장



(그림 2) 쓰기 기능을 위한 *ext4_file_write_iter* 함수 수정

쓰기 함수가 호출되면 *VB*자료구조 참조해 가변 길이 블록의 여부를 판별한다. 이 기능을 적용하기 위해 *ext4_file_write_iter()*함수를 수정하였다. 함수가 호출되면 가장 먼저 Inode의 정보를 불러온 후, 이 구조체 내에 VB 존재 여부와 인덱스를 참조해 가변 길이 블록이 적용된 블록인지 판별한다. 편의상 본 시스템에서는 4KB로 데이터가 처리된다고 가정하였다. 만약 가변길이 블록일 경우 *__generif_file_write_iter()*함수를 통해 페이지에 데이터를 복사하고, 그렇지 않을 경우 다음에 쓸 위치만 이동시킨다. 쓰기 함수도 이와 유사하게 작동하며, 파일을 읽는 과정에서 해당 블록이 가변 길이 블록인지 판별하고, 가변 길이 블록일 경우 범퍼영역을 제거한 후 유저 어플리케이션으로 복사한다.

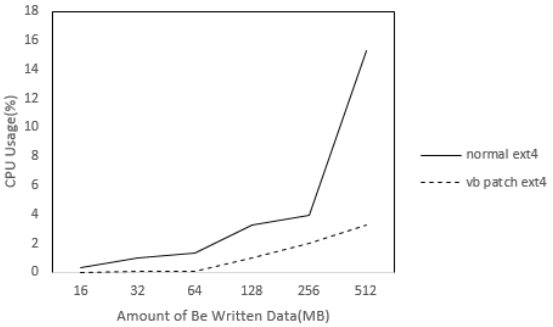
4. 실험 및 성능평가

본 논문에서 제안하는 시스템 프로토타입인 관계로 실험용 프로그램을 작성해 일부 적은 양의 데이터를 수정한 후 다시 저장할 수 있도록 하였고, 파일의 내용은 *dd*명령어를 통하여 임의로 생성하였다. 또한 *atop*과 *top*명령어를 사용하여 쓰기 연산양과 CPU사용량을 측정하였다.

표 1은 파일 크기별로 데이터를 일부 수정하고 다시 저장한 후의 모습을 보이고 있다. 가변 길이 블록의 개념을 적용할 경우는 4KB단위로 수정된 블록을 찾아 다시 저장하므로 파일 크기와 상관없이 모두 4KB의 쓰기 양을 보이고 있다.

<표 1> 파일 크기에 따른 데이터 수정 후 다시 저장 시 쓰기 연산 양

size(MB)	normal ext4(MB)	vb patch ext4(MB)
16	16.384	0.004
32	32.768	0.004
64	65.536	0.004
128	128	0.004
256	256	0.004
512	513.576	0.004



(그림 3) 파일 수정 시 CPU사용량

가변 길이 블록 개념을 적용할 경우 써지는 블록마다 수정이 발생된 블록인지 확인해야 하므로 추가적인 연산이 발생할 것으로 보인다. 하지만 실제로는 수정이 발생하지 않은 대다수의 블록에 대해 추가적인 커널 함수들을 수행하지 않으므로 적은 CPU 사용으로도 쓰기 연산을 완료할 수 있음을 그림 3에서 확인할 수 있다.

5. 결론 및 향후 연구 계획

본 논문에서는 유저 레벨에 적용됐던 가변 길이 블록 파일 시스템을 커널 레벨에서 구현하기 위해 ext4 파일시스템을 수정하였으며, 유저 어플리케이션에서 수정된 정보를 커널로 전달하기 위한 별도의 시스템 콜을 추가하였다. 비록 아직은 프로토타입으로써 수정위치 탐색, 시스템 콜 매개변수 전달, 4KB의 블록크기 고정 등 제한사항이 존재하지만 수정된 블록만 다시 디스크에 저장함으로써 적은 쓰기 양으로 파일을 수정할 수 있었다.

향후 연구에서는 앞서 언급한 문제점들을 해결함으로써 보편적인 파일시스템에 가변 길이 블록의 개념을 적용할 수 있도록 개선할 예정이다.

참고문헌

[1] A. Mathur, M. Cao, S. Bhattacharya, A. Dilger, A. Tomas, and L. Vivier, "The new ext4 filesystem: current status and future plans," in *Proceedings of the Linux Symposium*, pp. 21-33, 2007.

[2] S. Brin, J. Davis, and H. Garcia-Molina, "Copy detection mechanisms for digital documents," in *ACM SIGMOD Record*, pp. 398-409, 1995.

[3] 유영준, 김병관, and 고영웅. "가변 길이 블록을 이용한 유저 레벨 파일 시스템 설계 및 구현." 한국정보과학회 학술발표논문집 (2015): 1677-1679.