Open ROW Snapshot 파일시스템

석진선[O] 김문경[1] 노재춘[1] 박성순[2]
세종대학교[O] 세종대학교[1] 안양대학교[2]
durgatm@gmail.com, kmk1030@naver.com, jano@sejong.ac.kr, sspark@anyang.ac.kr

Open ROW snapshot file system

Jinsun Suk[O], Moonkyung Kim[1], Jaechun No[1], Sung Soon Park[2]
Sejong University[0], Sejong University[1], Anyang University[2]

요 약

스냅샷은 파일 시스템의 손상 또는 사용자의 부주의로 인한 데이터 손실을 방지하기 위한 기술로 크게 볼륨 단위로 스냅샷 이미지를 생성하는 방법과 파일 단위로 스냅샷 이미지를 생성하는 방법으로 나뉜다. 첫 번째 방법은 현재 널리 사용되고 있는 방법으로 스냅샷 이미지를 생성하는데 걸리는 시간이 짧고 디스크 관리가 용이하다는 장점과 생성된 이미지의 개수와 파일 시스템의 I/O 성능이 반비례한다는 단점을 가지고 있다. 두 번째 방법은 다수의 스냅샷 이미지를 생성한 후에도 파일 시스템의 I/O 성능이 저하되지 않는다는 장점과 스냅샷 이미지를 생성하는데 걸리는 시간이 파일의 개수와 비례한다는 단점을 가지고 있다.

본 논문에서는 기존의 파일 시스템 단위의 스냅샷 기능이 가지는 단점을 극복하기 위한 Open ROW Snapshot에 대해서 언급한다.

1. 서 론

다양한 스토리지와 파일 시스템들은 데이터의 신뢰도와 효율성을 향상시키기 위해 스냅샷을 이용한 데이터 버전닝을 사용하고 있다.

리눅스에서의 스냅샷은 스냅샷 이미지의 생성 단위를 기준으로 분류한 볼륨단위의 스냅샷 기법 파일 단위의 스냅샷 기법과 블록을 관리하는 기법을 기준으로 분류한 COW(Copy On Write)와 ROW(Redirect On Write)로 분류할 수 있다. 본 논문에서는 파일 단위로 스냅샷을 생성하는Open ROW Snapshot 파일 시스템에 대해서 언급한다. Open ROW Snapshot은 XFS를 기반으로 설계된 파일 시스템으로 ROW 방식의 스냅샷 기능 제공, 스냅샷 이미지의 생성 시간 단축 그리고 기반 파일 시스템으로 사용된 XFS의 I/O 성능을 최대한 보존하는 것을 목표로 설계, 구현된 파일 시스템이다.

2. 알고리즘

Open ROW Snapshot은 크게 세 가지 알고리즘을 사용한다. 첫 번째와 두 번 째 알고리즘은 스냅샷을 생성하고 원본 파일을 유지하기 위한 알고리즘으로 스냅샷 이미지를 생성하고 사용자에 의해 파일이 수정되거나 삭제되는 경우에도 원본 파일을 유지하기 위해 사용되고 세 번째 알고리즘은 생성된 스냅샷 파일을 특별한 절차 없이 안전하게 삭제하기 위한 알고리즘이다.

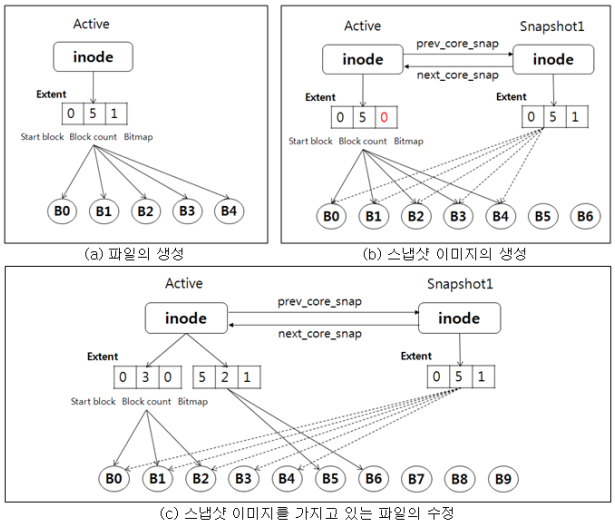

그림 1 Open ROW Snapshot의 Redirect 알고리즘

2.1 스냅샷 알고리즘

Open ROW Snapshot은 파일 시스템 단에서 수행되는 스냅샷 기법으로 생성하고자 하는 스냅샷 파일의 개수와 스냅샷을 생성하는데 걸리는 시간이 비례하는 단점을 극복하기 위해서 생성될 스냅샷 이미지를 위한 아이노드 구조체를 미리 할당해 놓는 방법을 사용하였다. 다시 말해서 파일이 생성되는 시기에 파일의 정보를 담은 두 개의 아이노드가 생성되고 이중 하나의 아이노드는 스

"본 논문은 Seoul R&BD Program(10557) 지원을 받아 수행되었음"

냅샷 이미지를 생성하는데 사용하여 스냅샷 이미지를 생성하는데 걸리는 시간을 최소한으로 줄이고자 했다.

2.2 Redirect 알고리즘

[그림 1]는 Open ROW Snapshot 파일 시스템에서 동작하는 스냅샷을 단계별로 묘사한 것이다. [그림 1]의 (a)는 파일을 생성했을 때의 모습을 표현한 것으로 파일은 B0~B4까지의 블록들로 이루어져 있으며 비트맵 값으로 "1"을 가지고 있다. 비트맵 값 "1"은 블록이 새로 생성되었음을 나타내는 것으로 공유되고 있지 않음을 의미한다. [그림 1]의 (b)는 (a)에서 생성했던 파일의 스냅샷을 생성한 후의 모습을 표현한 것이다. 스냅샷을 생성한 직후의 스냅샷 파일과 원본 파일은 동일한 메타데이터를 유지하여 데이터의 복사 없이 동일한 데이터에 접근하는 것이 가능하다. 활성(active) 파일의 비트맵 값이 "0"으로 바뀐 것은 snapshot1 파일의 생성으로 인해 블록이 공유되고 있음을 나타내기 위한 것이다. 이러한 경우엔 기존의 데이터를 보존하기 위해 사용자가 수정한 활성 파일의 데이터를 새로운 블록을 할당하여 기록해야할 필요가 있다. 활성화 파일과 활성화 파일에 대한 스냅샷 파일들은 더블 링크드 리스트로 연결되어 관리되며 이에 대한 자세한 언급은 스냅샷 파일의 삭제 알고리즘에서 언급할 것이다. [그림 1]의 (c)는 활성화 파일의 스냅샷 이미지를 생성한 후에 활성화 파일을 수정한 결과를 표현한 것으로 B3과 B4의 비트맵 값이 "0" 이므로 새로운 블록인 B5와 B6을 할당하고 수정된 내용을 기록하여 스냅샷 이미지의 데이터를 보존한 것을 볼 수 있다. 활성화 파일의 B5와 B6은 새로 할당된 블록으로 수정 되어도 스냅샷 이미지에 영향을 미치지 않으므로 비트맵 값으로 "1"을 가진다.

2.3 삭제 알고리즘

삭제 알고리즘도 extent의 비트맵 값을 이용하여 삭제 가능 여부를 판단한다. Extent의 비트맵은 "0" 또는 "1" 값을 가진다. 삭제 알고리즘은 비트맵 값에 따라 다르게 동작한다. 비트맵 값이 "0"인 경우는 prev_core_snap으로 연결된 파일과 블록을 공유한다는 의미이기 때문에 블록을 삭제할 수 없다. 반면에 비트맵 값이 "1"인 경우는 next_core_snap에 의해 공유될 가능성이 있으므로 next_core_snap의 비트맵 값에 따라 처리해야 한다. next_core_snap의 비트맵 값이 "0"인 경우는 블록이 공유됨을 의미하므로 삭제할 수 없고 "1"인 경우엔 공유되지 않음을 의미하므로 삭제하는 것이 가능하다.

4. 성능 측정

Open ROW Snapshot의 읽기/쓰기 성능은 IOzone[12]을 이용하여 측정하였으며 스냅샷 이미지 생성의 성능증가를 위해 할당한 두 개의 inode로 인한 성능저하가 측정되었다. 하지만 스냅샷 이미지를 생성하는데 필요한 시간은 스냅샷 이미지 생성 시에 inode를 할당하는 기법과 비교했을 때 우수한 성능을 보였다.

5.결론

OpenROWSnapshot 파일 시스템은 스냅샷 생성 시간을 단축하기 위해 개선된 스냅샷 알고리즘을 사용하였다. 이러한 시도는 기존의 파일 단위 스냅샷보다 뛰어난 성능을 보였지만 여전히 파일의 개수와 스냅샷 생성 시간이 비례한다는 단점을 가지고 있다.

본 논문에서 제안한 스냅샷 알고리즘은 앞으로 파일 단위의 스냅샷을 제공하지만 파일 시스템에 종속적이지 않은 모습으로 개선될 계획이다.

6.참고문헌

1. A.~Sweeney,D.~Doucette, W.~Hu, C.~Anderson, M.~Nishimoto and G.~Peck, "Scalability in the XFS File System", USENIX 1996: Annual Technical Conference, 1996.
2. XFS, "http://oss.sgi.com/projects/xfs/"
3. J.~Mostek, W.~Earl and D.~Koren. "Porting the SGI XFS File System to Linux", 6th Linux Kongress: The Linux Storage Management Workshop (LSMW), 1999.
4. Z. N. J. Peterson, R. Burns. "Ext3cow: The design, implementation, and analysis of metadata for a timeshifting file system." : Technical report, Hopkins Storage Systems Lab., Department of Computer Science, The Johns Hopkins University, 200

12. IOzone, "http://www.iozone.org"