NTFS 파일 시스템 복구 구현

김순환*, 김한규
홍익대학교 컴퓨터공학과
kcircle*@cs.hongik.ac.kr, hkim@cs.hongik.ac.kr

NTFS File System Recovery

Soon Hwan Kim*, Han-gyoo Kim
Dept. of Computer Eng., Hong Ik Univ.

요 약

인터넷의 확산으로 인한 바이러스의 감염, 정전 등의 재해로 인해 파일 시스템이 손상될 수 있기 때문에 이를 복구하기 위한 기법들에 대한 다양한 연구가 이루어지고 있다. 그러나 기존 연구들은 파일 시스템이 정상적으로 동작하는 환경에서 수행 도중의 오류를 복구하기 위한 로그 기법들이 주로 사용되고 있다. 이러한 기법들은 파일 시스템의 기본 구조가 손상될 경우 해당 디스크에 접근이 불가능하기 때문에 한계점이 존재한다. 따라서 본 논문에서는 파일 시스템 기본 구조 복구 모델을 제안한다. 또한 실수로 파일을 완전히 삭제할 경우 운영체제가 제공하는 정상적인 방법으로는 삭제된 파일에 접근 할 수 없다. 따라서 완전히 삭제된 파일을 복구할 수 있는 방안을 제시하고 이를 구현하였다.

1. 서 론

컴퓨터 환경이 지속적으로 발전함에 따라 사용되는 데이터의 양이 증가하게 되었고 그에 따른 저장장치의 크기도 함께 증가해왔다. 그러나 바이러스 감염이나 정전과 같은 예상치 못한 상황으로 인해 파일 시스템이 손상되는 경우가 빈번하게 발생한다. 또한 손상된 파일 시스템을 복구하기 위해서는 많은 시간이 걸리고, 때에 따라서 복구하지 못하는 경우도 발생한다. 따라서 파일 시스템의 복구가 중요한 이슈가 되어 왔으며, 이와 관련한 해결 방안들이 연구되어 왔다.[1, 2]. 그러나 지금까지의 연구들은 로그를 이용하여 파일 시스템이 정상적으로 사용 가능한 환경에서 발생하는 문제에 대해 복구하는 것에 대한 것이 대부분이기 때문에 파일 시스템 자체에 접근할 수 없는 경우에는 복구가 불가능 하다는 단점이 있다. 따라서 이러한 문제를 해결하기 위한 파일 시스템 기본 구조의 복구에 대한 연구가 필요하다.

본 논문에서는 Windows 환경에서 주로 사용되는 NTFS에서 파일 시스템에서 기본 구조가 손상되었을 경우의 복구 방안을 제시한다. 그리고 완전히 삭제된 파일의 복구에 대한 방안을 제시하고 이를 구현한다.

본 논문은 다음과 같이 구성되어 있다.

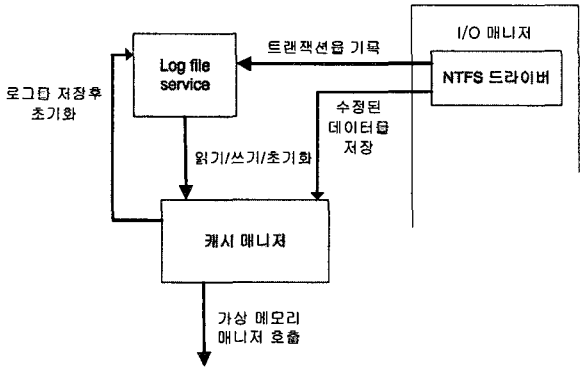
2장에서는 NTFS에서 제공하는 복구와 관련된 기존 연구 내용과 그 한계점을 기술하고, 3장에서는 본 논문에서 제안한 파일 시스템 기본 구조의 복구 방안과 완전히 삭제된 파일의 복구 방안에 대하여 설명한다. 4장에서는 본 논문의 결론 및 향후연구를 제시한다.

2. 관련 연구

2.1. NTFS에서의 복구

NTFS는 "Frangipani"등의 파일 시스템[1,2]들과 같이 로그를 이용하여 복구를 수행하고, LFS(Log File Service)[3]라는 커널 내 서비스를 이용하여 모든 데이터를 기록한다.

<그림 1>에서의 NTFS 드라이버는 파일을 수정하는 경우 트랜잭션을 LFS에 전송하고 수정된 데이터를 캐시 매니저를 통해 저장하는 모습을 나타낸다.



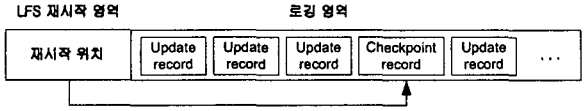
<그림 1> Log file service

예를 들면 파일 이름의 수정과 같은 작업은 파일 이름이 사용될 공간의 할당, 파일 이름의 수정 등의 세부 작업이 모여 하나의 트랜잭션을 이루게 된다. 이러한 트랜잭션은 트랜잭션내의 모든 작업을 모두 실행되게 하거나, 모두 실행되지 않도록 한다.

공간의 할당, 정보의 수정 같은 각각의 작업들은 '사용될 공간의 할당/사용될 공간의 해제'와 같이 수행에 필요한 정보와, 취소에 필요한 정보들을 포함하는 업데이트 레코드로 구성된다[3]. 또한 매 5초마다 주기적으로 체크포인트 레코드를 저장하여 복구시 어디서부터 시작할지를 알 수 있게 해준다. 그 결과 <그림 2>와 같은 로깅 영역이 만들어지게 되고 이 정보를 바탕으로 해당 파티션이 활성화될 때, 이전에 완료되었으나 실제 디스크에 적용되지 않은 트랜잭션을 재실행 하거나 완료되지 않은 트랜잭션을 취소한다.

이와 같이 로그를 이용한 파일 시스템 복구는 정상적인 파일 시스템 동작 중에 나타날 수 있는 오류를 복구한다. 그러나 파일 시스템 기본 구조가 손상될 경우 디스크 자체에 접근할 수 없기 때문에 파일 시스템이 정상적으로 동작 할 수 없고, 로그를 이용한 복구가

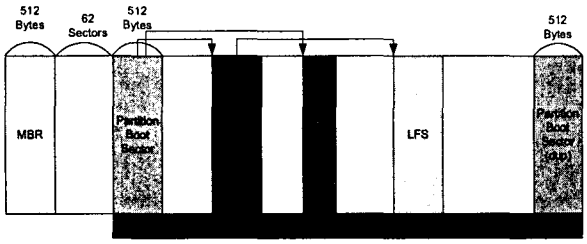
불가능하게 된다. 따라서 파일 시스템 기본 구조의 손상을 위한 복구 방안에 대한 연구가 필요하다.



<그림 2> LFS 로그 저장

3. NTFS 기본 구조의 복구

3.1. NTFS 기본 구조



<그림 3> 하나의 NTFS 파티션을 갖고 있는 디스크의 전체구조

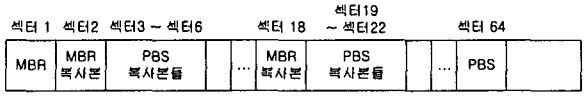
디스크는 <그림 3>과 같이 맨 앞에 MBR(Master Boot Record)이 존재한다. MBR은 디스크내의 파티션에 대한 정보를 포함하고 있고[4,5], 최대 4개의 파티션이 만들어 질 수 있다. 각 파티션은 확장 파티션이 되어 다른 논리 드라이브로 새로운 파티션을 설정할 수 있다. 이와 같이 MBR내에 파티션 정보가 존재하기 때문에 MBR이 손상될 경우 디스크 안의 구조를 파악할 수 없다. 따라서 MBR은 디스크에 접근하기 위한 필수적인 데이터라고 할 수 있다. MBR은 첫 번째 섹터에 위치하며 그 다음에 위치하는 62개의 섹터는 사용되지 않는 영역이다. 따라서 이곳에 MBR의 복사본을 저장할 수 있다. 본 논문에서는 MBR 다음의 62 섹터를 파일 시스템 기본 구조의 복구에 이용한다.

그리고 파티션의 첫 부분에는 PBS(Partition Boot Sector)가 위치하는데, PBS에는 부팅시 필요한 파일 테이블의 위치가 기록된다. 부팅 과정에서 파일 테이블이 손상될 경우를 대비하여 파일 테이블의 복사본의 위치가 부가적으로 기록된다. 또한 PBS의 복사본은 해당 파티션의 제일 마지막 섹터에 복사되어 있다. 본 논문에서는 PBS의 새로운 복사본을 MBR 다음 62개의 빈 섹터에 추가로 기록하여 PBS의 손상에 신속히 대처할 수 있도록 한다.

MFT(Master File Table), MFT Mirr, LFS(Log File Service)는 로그 기반의 복구 방법에 의해 관리되는 정보이므로 본 논문에서는 사용하지 않는다. MFT는 해당 파티션에 존재하는 파일 정보를 저장하고 관리하는 파일 테이블이다. MFT Mirr는 MFT의 손상에 대비하여 중요 파일 엔트리를 따로 저장하고 있다. LFS는 로그 정보를 저장하기 위한 장소이다.

3.2. 기본 구조 복구 모델

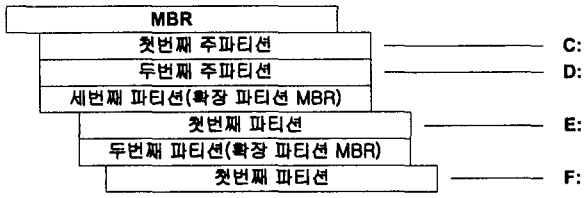
<그림 4>는 MBR 다음에 위치하는 62개의 빈 섹터에 중요 기본 구조 데이터인 MBR, PBS이 저장된 것을 나타낸 것이다. 두 번째 섹터에 MBR의 복사본을 저장하고, 그 다음 4개의 섹터에 MBR이 포함하고 있는 파티션의 PBS를 순서대로 저장한다. 또한 MBR의 복사본이 손상되거나 파티션의 잘못된 조작이 있을 경우를 대비하여 두 번째 복사본도 저장한다. 특히 연속된 섹터일 경우 동시에 손상될 가능성이 있으므로 첫 번째 복사본이 저장될 위치에서 16개의 섹터가 떨어진 위치에 두 번째 복사본을 저장한다.



<그림 4> 복구를 위한 저장구조

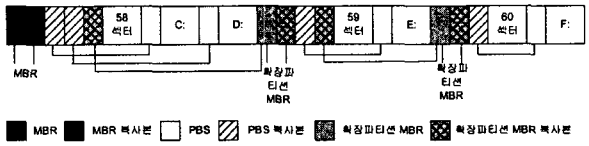
3.3. 기본 구조 복구 모델의 확장

디스크의 각 파티션은 확장파티션이 될 수 있고, 각 확장파티션은 하나의 논리 드라이브와 하나의 확장 파티션으로 될 수 있다. 따라서 하나의 디스크는 확장파티션에 의해 여러 개의 파티션으로 구성될 수 있다.



<그림 5> 확장 파티션의 계층 구조

<그림 5>는 두 개의 주 파티션과 두 개의 확장 파티션으로 구성된 디스크를 나타내고 있다. <그림 5>의 구조에서는 상위 계층의 확장 파티션 MBR이 손상되었을 경우 하위 파티션을 접근하지 못하게 된다. 따라서 확장파티션 MBR의 다음에도 62개의 섹터가 사용되지 않고 비어있기 때문에 이곳에 확장 파티션의 MBR을 저장하여 문제를 해결할 수 있다.



<그림 6> 복구를 위한 저장구조의 확장

<그림 6>는 <그림 5>의 파티션 구조에서 파일 시스템 기본 구조 복구를 위해 각 데이터들이 저장된 후의 모습이다. 단, 두 번째 복사본들은 저장되지 않고 첫 번째 복사본들만 저장된 상태를 보여준다.

3.4. 기본 구조의 복구 순서

디스크가 활성화될 때 <그림 7>의 알고리즘에 의해 복사본을 생성한다. 디스크의 파티션 구성에 따라 <그림 6>과 같은 저장구조가 만들어지게 된다.

디스크나 파티션이 손상되어 인식이 불가능할 경우 다음과 같은 순서로 복구 프로그램을 수행한다. 원본 MBR, PBS가 손상이 되었을 경우에는 첫 번째 복사본을 가져와 복구를 수행한다. 이 때, 잘못된 파티션 조작으로 인한 경우에는 복구할 파티션 데이터를 선택해야 한다. 파티션 조작 후 첫 번째 복사본과 두 번째 복사본의 데이터가 변경될 수 있기 때문에 사용자가 복구될 파티션 정보를 선택해야

한다. 파티션 테이블만을 수정했을 경우 모든 파일 시스템 기본 구조 및 파일 시스템을 복구할 수 있으나, 파티션 조작 후 파티션을 초기화했거나 파티션 내의 섹터들을 수정했을 경우 이전 상태로 완벽하게 복구하지 못할 수도 있다.

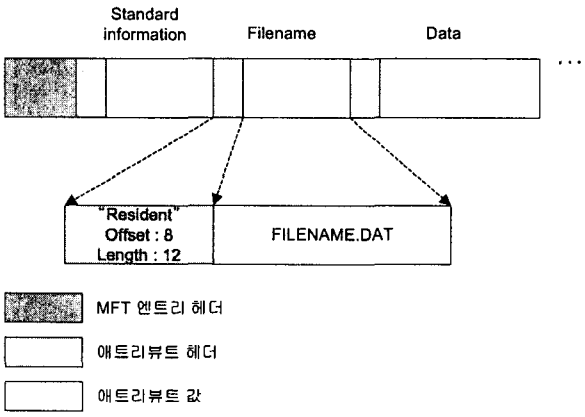
```
if (MBR과 두 번째 섹터의 데이터가 다르다)
  두 번째 섹터의 데이터를 18번째 섹터에 저장한다.

MBR섹터를 두 번째 섹터에 저장한다.

loop(파티션이 존재한다) {
  if (PBS/확장 파티션 MBR과 저장될 위치의 데이터가 다르다)
    저장될 위치의 데이터를 (현재 위치 + 16)의 섹터에 저장한다.
  파티션의 PBS/확장 파티션 MBR의 복사본을 저장한다.
}

if (MBR내에 확장 파티션이 존재한다)
  확장 파티션 MBR을 MBR로 생각하고 처음부터 다시 수행한다.
```

<그림 7> 파일 시스템 기본 구조 복구를 위한 알고리즘

3.5. 파일 데이터의 복구

파일을 완전히 삭제하는 경우 운영체제가 제공하는 정상적인 작업으로는 파일의 접근이 불가능하다. 그러나 NTFS는 파일을 완전히 삭제할 경우 MFT내의 파일 엔트리에 파일이 삭제되었음만을 표시한다. 따라서 삭제된 파일의 정보가 있는 섹터가 다른 파일에 의해 손상되지 않았다면 완전히 삭제한 파일이라도 디스크에 직접 접근하여 복구할 수 있다.

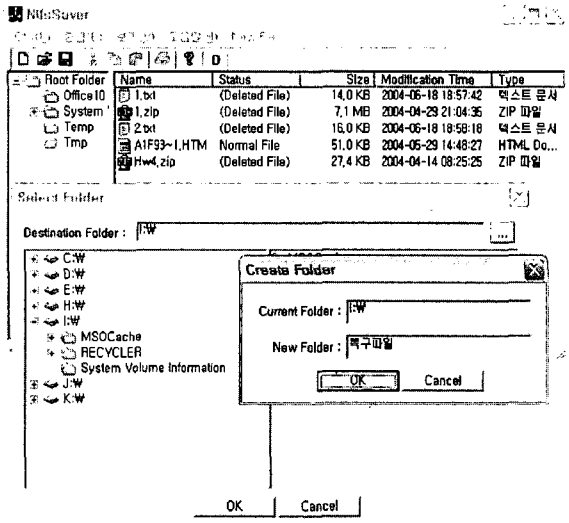
파일의 기본적인 정보는 MFT내 각 엔트리에 <그림 8>과 같이 애트리뷰트의 형태로 생성날짜, 파일 이름 같은 데이터가 저장된다. 따라서 삭제된 파일을 복구할 때 애트리뷰트 형태로 저장된 파일 이름과 데이터를 알아낼 수 있다.



<그림 8> 데이터가 애트리뷰트 형태로 저장되는 MFT 엔트리

본 논문에서는 탐색기 형태로 삭제된 파일 목록을 MFT에서 검색하고, 검색된 목록에서 복구할 파일을 선택하여 복구를 수행한다. 이 때, 같은 파티션에 복구된 파일을 저장할 경우 다른 파일 데이터에 영향을 줄 수 있으므로 다른 디스크에 저장을 해야 한다.

디렉터리의 경우에도 파일과 동일한 방법으로 복구를 수행한다. 그러나 디렉터리의 경우에는 디렉터리에 속해있는 파일의 목록을 애트리뷰트에 포함하고 있기 때문에 이곳에 저장되어 있는 파일 목록과 이 목록에 들어있는 파일들도 복구해야 한다. 그러나 해당 디렉터리에서 파일을 삭제할 경우, 그 디렉터리의 파일 목록에서 파일이 삭제되기 때문에 디렉터리를 복구할 경우 전체 파일을 복구할 것인지 삭제할 당시에 속해있던 파일들만을 복구할 것인지를 선택하도록 한다. <그림 9>는 파티션의 MFT를 검색하여 삭제된 파일을 복구하는 과정이 수행되는 실제 모습을 보여준다.



<그림 9> 완전히 삭제된 파일 데이터의 복구

4. 결론 및 향후 연구

파일 시스템의 MBR, PBS와 같은 기본 구조가 손상될 경우 디스크를 인식하지 못하거나, 저장되어 있는 파일에 접근할 수 없다. 따라서 저장된 파일 데이터의 복구뿐만 아니라 파일 시스템 자체를 구성하는 기본 구조의 복구 방안이 필요하다.

따라서 본 논문에서는 NTFS 기본 구조 복구 모델을 정의하고 기본 구조의 복구 방안과 완전히 삭제된 파일/디렉터리의 복구 방안을 제시하고, 구현하였다.

향후에는 로컬 환경에서 사용되는 디스크뿐만 아니라 네트워크에 접속되어 여러 시스템에서 동시에 사용되는 디스크의 신뢰성 구축 및 복구에 대한 연구가 필요하다.

<참고문헌>

[1] Chandramohan A. Thekkath et al, "Frangipani: A Scalable Distributed File System", Proc.of the 16th ACM SOSP, 1997

[2] Mendel Rosenblum and John K. Ousterhout. "The design and implementation of a log-structured file system.", ACM TCS, 1992

[3] David A. Solomon, Inside Windows NT SE, 1998

[4] NTFS.com, http://www.ntfs.com/

[5] Linux-NTFS Project, http://linux-ntfs.sourceforge.net/