경량 클라우딩을 위한 DB 기반 가상 파일 시스템 설계

이형봉*
*강릉원주대학교 컴퓨터공학과
e-mail:hblee@gwnu.ac.kr

Design of a DB-based Virtual File System for Lightweight Clouding

Hyung-Bong Lee*
*Dept of Computer Science & Engineering, Gangneung-Wonju National University

요 약

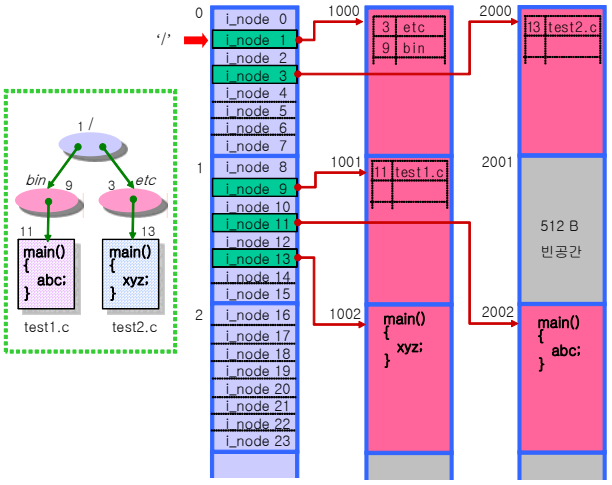
윈도우나 리눅스 등의 일반적인 파일 시스템은 데이터를 로컬 저장 장치에 직접 저장하고 관리하기 때문에 대용량 저장 공간을 기반으로 운영된다. 반면에, 클라우딩을 위한 가상 파일 시스템은 데이터를 로컬 저장 장치에 직접 저장하는 경우보다는 지역적으로 분산된 데이터에 대한 지시자만을 관리하는 경우가 더 많다. 이 연구에서는 소규모 클라우딩 환경에서 파일 시스템의 디렉터리 체계를 DB로 유지하면서 모든 데이터 파일들은 지시자로만 관리하는 유닉스 파일 시스템 스타일의 가상 파일 시스템을 설계하여 제안한다. 시험 결과, 트리 구조의 디렉터리 체계가 의도대로 이루어짐이 확인되었다.

1. 서론

클라우드 컴퓨팅의 일반적인 개념은 사용자가 언제 어디서나 용이하게 구성이 가능한 컴퓨팅 자원들의 공유된 풀에 요구 즉시 네트워크 접근이 가능한 모델인데, 이를 이용한 서비스는 소프트웨어형(SaaS: Software as a Service), 플랫폼형(PaaS: Platform as a Service), 기반구조형(IaaS: Infrastructure as a Service) 등 크게 3 가지로 분류된다[1]. 이들 서비스는 가상화의 기본 개념을 바탕으로 물리적 컴퓨팅 자원에서부터 소프트웨어 자원까지 점차 추상화 정도가 높아진다. 현재 일반 사용자들이 가장 실감나게 체험하고 있는 클라우드 서비스 중의 하나는 각종 인터넷 포털 사이트의 이메일 서비스를 들 수 있다. 이를테면 Google의 G메일은 지역적으로 분산된 엄청난 규모의 저장 공간을 가상 파일 시스템으로 구축하여 사용자들에게 메일링 소프트웨어 서비스를 제공하고 있다[2]. Dropbox는 PC나 스마트폰 구별 없이 인터넷상의 저장 공간을 가상 파일 시스템으로 제공함으로써 인터넷이 있는 곳이면 어디에서도 동일한 파일 시스템 뷰(view)로 접근할 수 있다[3]. 이 논문에서는 중견 기업 정도의 소규모 클라우딩 환경에서 지역적으로 분산된 실제 파일들을 하나의 논리적 파일 시스템으로 구축하는 방안의 하나로써, 디렉터리 체계를 DB에 유지하면서 개별 파일들을 지시자(포인터)로 관리하는 유닉스 스타일의 경량 가상 파일 시스템을 설계한다.

2. 유닉스 파일 시스템

유닉스 파일 시스템은 그림 1과 같이 디렉터리를 포함한 모든 파일에 대하여 유일한 i-node를 할당하고, i-node에는 해당 파일의 내용(데이터)을 저장하고 있는 블럭들의 번호(지시자)를 기록한다[4]. 디렉터리 파일의 경우 그 디렉터리에 연결된 하부 파일들의 i-node 및 파일명으로 구성된 테이블을 파일 내용으로 보관한다. 디렉터리 파일의 내용은 트리 구조 형태의 파일 시스템 체계를 유지하는 근간이므로, 일반 사용자는 디렉터리 내용을 직접 수정할 수 없고 반드시 운영체제 서비스를 통하여 수정하여야 한다. 이를테면 사용자가 “/etc” 아래에 ”net”라는 디렉터리를 생성하려면 mkdir(”/etc/net”) 형태의 서비스를 호출한다.



(그림 1) 유닉스 파일 시스템

3. DB 기반 경량 가상 파일 시스템

그림 1의 디렉터리 파일에는 다른 파일에 대한 지시자들만 관리되므로 그 크기가 크지 않아 DB로 구현하더라도 성능상 큰 문제는 일어나지 않는다. 디렉터리 체계를 DB에 관리하면 운영체제 커널이 아닌 사용자 영역에서 구현할 수 있다는 큰 장점을 얻을 수 있다.

□ 가상 파일 시스템을 위한 DB 스키마

유닉스 파일 시스템 보안 모델은 사용자 그룹("/etc/group"), 사용자 계정("/etc/passwd")에 정의된 그룹 및 사용자 id에 근간을 두고 있다. 이 연구에서는 이들 두 파일과 그림 1의 파일 시스템 체계를 DB 테이블로 구현하였는데, 이들 테이블 생성을 위한 MySQL 쿼리 문장을 그림 2에 보였다. 'vfs' 테이블의 각 레코드는 디렉터리 파일의 엔트리에 해당하고, 그들이 어느 디렉터리에 포함되는지는 pinode(parent inode)와 inode(child inode) 관계에 의해 결정된다. 또한, 해당 엔트리에 대한 디렉터리/파일의 식별은 'type' 필드로 이루어지고, 파일인 경우 해당 파일의 위치 정보를 별도의 필드에 기록한다.

```
create table vgroup (
      gname    varchar(8)   not null,
      gid      int unsigned primary key);
create table user  (
      uname    varchar(8)   not null,
      pass     varchar(8)   not null,
      uid      int unsigned primary key,
      gid      int unsigned not null,
      foreign key (gid)  references vgroup(gid));
create table vfs (
      pinode   int unsigned not null, // parent inode
      inode    int unsigned not null, // child inode
      name     varchar(15)  not null, // file name
      uid      int unsigned not null,
      gid      int unsigned not null,
      type     varchar(1)   not null,
      primary key (pinode, name),
      foreign key (uid) references vuser(uid),
      foreign key (gid) references vgroup(gid));
```

(그림 2) 가상 파일 시스템을 위한 DB 스키마

□ 가상 파일 시스템 운영 프로시저

파일 시스템 운영을 위한 핵심 프로시저는 "/usr/home/hblee/vfile" 형태의 경로명을 파싱하여 최종적으로 'vfile' 파일에 대한 i-node(디렉터리 엔트리)를 검색하는 것이다. 그림 2의 'vfs' 테이블은 생성 후, 파일 시스템 초기화 즉, 포맷 과정으로 (1, 1, "/", 0, 0, ...), (1, 1, ".", 0, 0, ...), (1, 1, "..", 0, 0, ...) 등 세 개의 엔트리가 등록되고, 이를 바탕으로 경로명을 파싱하는 프로시저는 그림 3과 같다. 그림 4에는 초기화 후 약간 확장된 'vfs' 테이블 내용과 find 형태의 프로시저로 탐색한 모습을 보였다.

4. 결론

이 연구에서는 가상 파일 시스템 운영의 기반이 되는 트리 구조의 디렉터리 체계를 DB로 관리하기 위한 스키마를 설계하고 시험하였다. 앞으로 mkdir(), rmdir(), unlink(), open(), close(), read(), write() 등의 파일 접근 프리미티브(시스템 서비스)를 목적에 부합하도록 구현하여 하나의 완전한 가상 파일 시스템으로 완성할 예정이다.

```
procedure int vfs_parse_path(in path,
                        out inode, out fname, out type) {
    cfname="/"; cinode=1; ctype = 'd';
    ctoken=get_next_token_from_path(path);
    if (cfname ≠ ctoken) return FAIL;
    while(TRUE) {
       ctoken = get_next_token_from_path(path);
       if (ctoken == NULL) {
           inode=cinode; fname=cfname; type=ctype;
           return SUCCESS;
       }
       pinode = cinode;
       cinode = vfs_get_dir_entry(pinode, ctoken, ctype);
       if (cinode == INVALID) return FAIL;
       pinode=cinode; cfname=ctoken;
    }
}
procedure int vfs_get_entry(in pinode, in cfname,
                                                out ctype) {
    search vfs table with pinode and cfname;
    return the search results(inode and type);
}
```

(그림 3) 가상 파일 시스템 경로명 검색 알고리즘

pinode	inode	fname	gid	uid	type
0	1	.	0	0	d
0	1	..	0	0	d
0	1	/	0	0	d
1	4	b	0	0	d
1	2	home	0	0	d
2	2	.	0	0	d
2	1	..	0	0	d
2	3	hblee	0	0	d
3	3	.	0	0	d
3	2	..	0	0	d
3	8	x	0	0	d
4	4	.	0	0	d
4	1	..	0	0	d
4	5	c	0	0	d
5	5	.	0	0	d
5	4	..	0	0	d
5	7	F	0	0	f
5	6	N	0	0	n
8	8	.	0	0	d
8	3	..	0	0	d

(그림 4) 가상 파일 시스템 구현 모습

참고문헌

[1] Su-Mi Shin, Mi-Whan Hyun, Hae-Sun Kim, "Trends and Cases of Cloud-based Information Service", Knowledge Report of KISTI, No. 39, December 2013, pp. 8-14(in Korean).

[2] Park So Yeon, Kim Yongwon. "An Analysis of the Interaction Effect of Benefit and", KIPS Transactions on Computer and Communications Systems, Vol. 2, No. 1, February, 2013, pp. 27-34(in Korean).

[3] Darren Quick, Kim-Kwang Raymond Choo, "Dropbox analysis: Data remnants on user machines", Digital Investigation, Vol. 10, No. 1, Februry 2013, pp. 3-18(in Korean).

[4] Hyung-Bong Lee, "UNIX Fundamental for Developers", Human Science, February 2012, pp. 396-403(in Korean).