관계형 데이터베이스를 이용한 계층적 파일 시스템의 메타데이터 관리 방법

김상완°, 곽재혁, 함재균, 황영철
한국과학기술정보연구원 슈퍼컴퓨팅센터 그리드컴퓨팅연구팀
{sangwan°, jhkwak, jaehahm, hychul}@kisti.re.kr

A Method for Managing Metadata of Hierarchical File System Using RDBMS

Sangwan Kim°, Jae-Hyuck Kwak, Jaegyoon Hahm, Youngchul Hwang
KISTI(Korea Institute of Science Technology Information) Supercomputing Center
Grid Computing Research Team

요 약

디렉터리와 파일의 계층적 구조를 가지는 계층적 파일 시스템은 오늘날 대부분의 범용 컴퓨터에서 흔히 사용되고 있다. 계층적 파일 시tm템은 직관적이고, 체계적이며, 단순하다는 장점이 있으나 검색이 용이 하지 않으며, 메타데이터를 관리하기 어렵다는 단점이 존재한다. 본 연구에서는 계층적 파일 시스템의 장점과 빠른 검색기능을 활용하여 메타데이터를 검색하고 관리할 수 있는 데이터베이스의 장점을 결합하여 계층적 파일 시스템에서 메타데이터를 관리할 수 있는 방법을 제안하였다. 데이터 그리드와 같이 분산된 데이터 저장 장치를 연동하여야 하는 경우에 원격지에 있는 파일 시스템의 파일들을 검색하는 일이 빈번히 수행되는데, 이 경우 본 연구에서 제안한 방법을 사용하면 효과적인 시스템을 기대할 수 있다.

1. 서 론

1.1. 계층적 파일 시스템

디렉터리 또는 폴더와 파일로 구성되는 계층구조를 가지는 계층적 파일 시스템(hierarchical file system[1])[1]은 컴퓨터 파일 시스템의 역사와 함께 현재까지 폭넓게 사용되고 있다. 오늘날 대부분의 범용 컴퓨터 시스템에서 계층적 파일 시스템의 구조를 이용하고 있다. 계층적 파일 시스템은 뿌리 디렉터리에서 부터 시작하여 하위 디렉터리가 계층적으로 분기되어 나간다. 각 디렉터리는 파일들과 다른 디렉터리를 포함할 수 있으며 뿌리 디렉터리에서부터 특정 파일까지의 디렉터리의 연쇄를 경로(path)라고 한다.

계층적 파일 시스템의 특징을 정리하면 다음과 같다:

- 파일들이 하향식 접근방식(Top-to-Bottom)으로 구성된다.
- 특정 데이터에 접근하기 위해서 뿌리에서 시작하여 계층 레벨을 따라 아래로 한 단계씩 찾아 들어간다.
- 디렉터리나 파일의 이름을 이용하여 원하는 자료를 찾는다.
- 경로(path) 또는 URL을 이용하여 파일의 유일한(unique) 이름을 나타낸다. (예, /Documents and Settings/username/My Documents/report.hwp)
- 디렉터리는 그 안에 들어 있는 파일과 하위디렉터리를 숨기는 역할을 한다.

직관적이고 체계적인 이러한 특징들 때문에 계층적 파일 시스템은 많은 컴퓨터 파일 시스템에서 사용되고 있다.

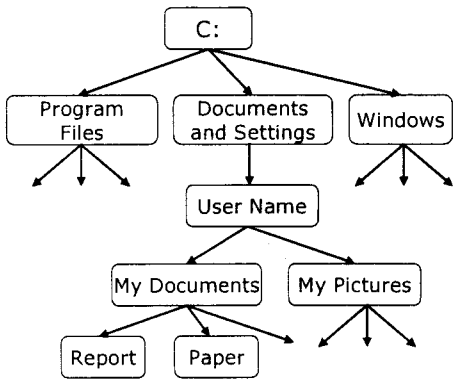


그림 1. 일반적인 Windows XP 파일시스템의 계층 구조

1.2. 계층적 파일 시스템의 단점

계층적 파일 시스템은 체계적이고 일관된 파일의 관리와 보관에 유리하다는 장점이 있지만, 파일의 분류 기준이 명확하지 않을 경우, 다시 말해서 파일을 분류하는 기준이 상호간에 계층적 관계를 가지고 있지 않을 경우에는 파일을

1) 참고로, 머리글자가 대문자로 표기된 Hierarchical File System은 애플 매킨토시 컴퓨터에서 사용되는 계층 구조 파일 시스템의 현행판을 가르킨다.

분류했을 때 유용성이 떨어지게 된다.[2] 당근과 오렌지를 예로 들어 보자. 당근과 오렌지는 둘 다 '먹을 수 있다'라는 특징과 '색깔이 오렌지색이다'라는 특징을 갖고 있다. 그러나 당근은 야채로 분류되고, 오렌지는 과일로 분류된다. 둘 다 유럽지역에서 생산되지만 당근은 주로 네덜란드에서, 오렌지는 주로 스페인에서 많이 생산된다. 당근과 오렌지를 분류하는 데 사용된 분류기준들 중 대부분은 기준들 간에 아무런 관계도 존재하지 않는다. 단지, 스페인과 유럽 사이에만 계층적 관계가 존재한다. 왜냐하면 스페인은 유럽의 일부분이기 때문이다. 계층적 파일 시스템에서는 파일을 분류할 때 파일이 가지고 있는 여러 속성들 중 한번에 한 가지씩만을 사용하여 분류를 해 나간다. 맨 첫 번째 분류 기준이 '과일이냐 야채냐'라는 기준이라면 찾으려고 하는 대상이 과일인지 야채인지 알아내기 전에는 다른 분류 기준을 적용하는 것이 불가능하게 된다.

또한 계층적 파일 시스템에서는 파일이나 디렉터리의 이름 이외에 파일의 내용에 관련된 메타데이터를 자체적으로 관리할 수 있는 방법을 제공하지 않는다. 파일의 이름으로만 파일의 내용을 추측 할 뿐, 파일의 내용에 대한 상세한 메타데이터가 필요하다면 파일 헤더를 메타데이터를 유지하기 위한 공간으로 활용하거나(그림2) 메타데이터를 보관하기 위한 별도의 파일을 만들어 관리해야 한다(그림3).

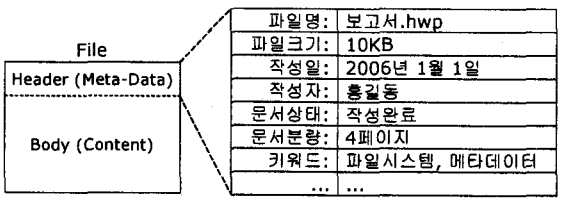


그림 2. 파일 헤더를 이용한 메타데이터 유지

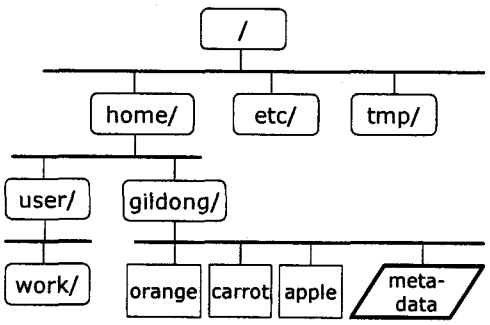


그림 3. 메타데이터 정보를 위한 별도의 파일 관리

이러한 방식은 모든 응용어플리케이션에 공통적으로 적용할 수 있는 방식이 아니며, 어플리케이션에 따라 메타데이터 저장 방법과 형식이 다르므로, 공통적이고 일관적인 메타데이터 관리 방법으로 볼 수 없다.

또, 계층적 파일 시스템에서는 검색이 용이하지 않다. 디렉터리 안의 파일 목록을 얻기 위해서는 디렉터리를 방문해야 하며, 한 번에 한 개의 디렉터리만 방문이 가능하기 때문에, 전체 파일 시스템 내에서 특정 파일을 검색하기위해서는 파일 시스템내의 모든 디렉터리를 순차적으로 모두 방문해야만 한다.

1.3. 메터데이터

메타데이터(Metadata)란 다른 데이터(흔히 이 데이터를 리소스(resource)라고 부름)를 설명하는 데이터이다.[4] 파일 시스템에서 사용되는 메타데이터는 용도와 응용프로그램 계층에 따라 여러 가지 종류로 나누어 볼 수 있다:

- 파일 레벨 메타데이터: 파일의 크기, 마지막으로 수정한 시간, 생성된 시간, 소유한 사람, 등등.
- 접근 제어관련 메타데이터: 파일의 접근허가(permision)과 관련된 데이터
- 응용프로그램을 위한 메타데이터: 응용프로그램에 의존적인 정보. 사용자가 필요에 따라 정의할 수 있는 정보. (예를 들어, 실험데이터의 경우 온도, 위도, 경도와 같은 실험 조건과 관련된 정보)

메타데이터를 이용하는 목적은 다음과 같은 것들이 있다:

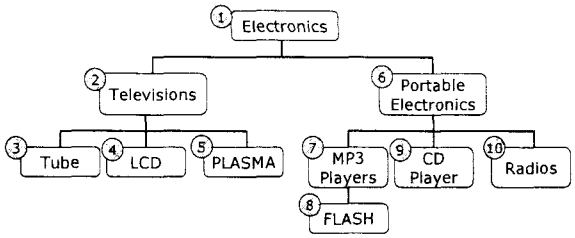
- 리소스를 구별하기 위해
- 관련된 정보를 쉽게 찾을 수 있도록 하기 위해
- 유사한 리소스를 함께 분류하기 위해
- 리소스의 위치를 알려주기 위해

본 연구에서는 앞에서 언급한 계층적 파일 시스템의 단점을 보완하기 위하여 계층적 파일 시스템의 메타데이터 정보를 관계형 데이터베이스(Relational Database Management System)를 이용하여 관리하는 방법을 제시한다. 2절에서는 파일 시스템의 계층 구조를 관계형 데이터베이스에 저장하는 방법과 더불어 메타데이터를 저장하고 관리하는 방법을 설명한다. 3절에서는 파일시스템에서 메타데이터를 표현하기 위해 속성-값 방식을 이용하여 관계형 데이터베이스로 표현하는 방법을 설명한다. 4절에서는 본 연구에서 제안된 방법으로 구현된 프로그램의 성능에 대해서 간략히 설명하였다. 마지막으로 결론에서는 본 연구의 필요성과 향후 연구 및 적용에 대해서 언급하였다.

2. 관계형 데이터베이스를 이용한 계층적 파일 시스템의 계층구조 표현

2.1. 계층구조를 RDBMS로 표현

관계형 데이터베이스(RDBMS)를 이용하여 계층적 구조를 표현하는 가장 간단한 방법은 디렉터리와 파일의 포함



(a) 계층구조

ID	Name	Parent
1	Electronics	0
2	Televisions	1
3	Tube	2
4	LCD	2
5	PLASMA	2
6	Portable Electronics	1
7	MP3 Player	6
8	FLASH	7
9	CD Player	6
10	Radios	6

(b) 인접목록을 이용하여 계층구조를 테이블로 표현함

그림 4. 인접 목록을 이용한 계층적 구조의 표현

관계를 계층적 트리(tree)구조에서 부모 노드와 자식 노드 간의 관계를 인접 목록(adjacency list)으로 표현하는 것이다.[3](그림 4) 계층구조에서 각각의 항목에 중복되지 않는 고유값을 할당을 하고, 부모-자식 간의 관계를 (부모 항의 고유값, 자식 항의 고유값)의 쌍으로 표현하는 방법이다.

2.2. 파일 시스템의 계층구조를 RDBMS로 표현

파일 시스템의 계층적 구조를 위와 같은 인접 목록으로 만들기 위해서는 파일이나 디렉터리를 대표하는 고유값을 정하고, 이 고유값으로 계층구조의 부모와 자식 간의 포함 관계를 표현하면 된다. Unix 또는 NTFS 파일 시스템에서 사용되는 i-node 값은 동일한 파티션 내에서 고유하기 때문에 인접 목록의 ID값으로 활용하기에 좋은 예가 된다. (그림 5)는 리눅스 운영체제에서 파일 시스템의 i-node와 인접목록을 이용하여 계층구조를 RDBMS로 표현한 예제이다.

3. RDBMS를 이용한 파일 시스템 메타데이터관리

관계형 데이터베이스 관리시스템은 메타데이터를 효과적으로 저장하고, 빠르게 검색할 수 있다는 장점이 있다.

RDBMS를 이용하여 파일 시스템의 메타데이터를 유지 관리하기 위해서는 다음과 같은 사항들이 필요하다:

- 표현할 수 있는 메타데이터의 종류에 제한이 없어야 한다: 메타데이터의 종류는 응용 용도에 따라 얼마든지

```
[root@host /]$ ls -li
17121281 drwxr-xr-x   57 root root  4096 Aug 21 20:29 etc
 6750209 drwxr-xr-x   23 root root  4096 Jul 19 09:49 home

[root@host /etc]$ ls -li
17122019 drwxr-xr-x   4 root root     4096 Apr 17 10:33 acpi
17121560 -rw-r--r--   1 root root       46 Jul 12 11:00 adjtime
17122473 -rw-r--r--   1 root root     4429 Aug  8 10:03 aliases
17122339 -rw-r-----   1 root smmsp  12288 Aug 21 20:28 aliases.db
17121333 drwxr-xr-x   2 root root     4096 May 17 10:16 alternatives
17121721 -rw-r--r--   1 root root      329 Apr 28  2005 anacrontab

[root@host /etc/acpi]$ ls -li
17122020 drwxr-xr-x  2 root root 4096 Apr 28  2005 actions
17122021 drwxr-xr-x  2 root root 4096 Apr 17 10:33 events

[root@host /etc/acpi/events]$ ls -li
17122022 -rw-r--r--  1 root root 90 Apr 28  2005 sample.conf
```

(a) 리눅스 파일시스템의 i-node 리스트

inode	parent inode	Name
2	0	/
17121281	2	etc/
6750209	2	home/
17122019	17121281	acpi/
17121560	17121281	adjtime/
17122473	17121281	aliases
17122339	17121281	aliases.db
17121333	17121281	altern/
17121721	17121281	anacrontab
17122020	17122019	actions/
17122021	17122019	events/
17122022	17122020	sample.conf
...	...	...

(b) i-node를 이용한 계층 구조의 표현

그림 5. 인접 목록을 이용하여 표현한 계층적 파일 시스템

다양화 될 수 있기 때문에 규격화되고 형식화 된 메타데이터뿐만 아니라 필요한 정보를 얼마든지 추가할 수 있어야 한다.

- 사용자가 정의한 메타데이터를 관리할 수 있어야한다: 사용자가 필요한 메타데이터를 새롭게 정의하여 관리(보관, 수정, 삭제, 검색)할 수 있어야 한다.

3.1. (속성, 값) 짝을 이용한 메타데이터의 표현

이와 같은 요구 사항을 만족시키기 위해서 본 연구에서는 (속성, 값)의 짝을 이용한 메타데이터의 표현 방법을 이용하였다. 특정 항목에 해당되는 정보를 (속성, 값) 형태로 표현하고, 이를 RDBMS의 각각의 칼럼으로 할당하여 저장하는 것이다.(그림 6)

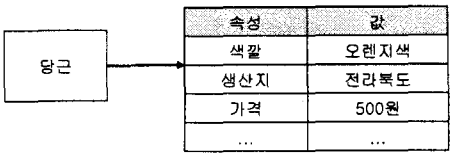


그림 6. (속성, 값)을 이용한 메타데이터의 표현

3.2. 메타데이터 저장을 위한 테이블 정의

(속성, 값)쌍이 어떤 항목의 속성인지 표시하기위해 메타데이터 정보 테이블을 아래와 같이 정의 한다:
- 고유값 (파일 시스템의 inode값)
- 속성 (문자열)
- 값 (문자열)

예를 들어, 실험데이터 파일과 그에 해당되는 속성을 그림 7과 같이 만들 수 있다.

inode	parent inode	name
2	0	/
1230000	2	home/
4560000	1230000	user/
7890000	4560000	실험결과/
7890001	7890000	결과1.data
7890002	7890000	결과2.data
...	...	...

(a) 계층구조를 표현한 테이블

inode	attribute	value
7890001	온도	80 C
7890001	속도	60km/h
7890001	실험자	홍길동
7890002	온도	70 C
7890002	속도	60km/h
7890002	실험자	김철수
...	...	...

(b) 메타데이터 정보 테이블

그림 7. 실험데이터에 대한 메타데이터의 표현 예

이 방법을 사용할 때의 장점은 다음과 같다:
- 한 항목에 관한 메타데이터를 개수에 제한 없이 표현할 수 있다: 하나의 항목에 대하여 속성을 서로 달리 함으로써 메타데이터 정보 테이블에 추가함으로써 특정 항목과 연관된 메타데이터를 제한 없이 기술 할 수 있다.
- 같은 속성 혹은 같은 값을 가진 항목을 빠르게 검색하 수 있다: (고유값, 속성, 값) 묶음으로 테이블에 보관되기 때문에 빠른 검색이 가능하다.
- 항목별로 연관되어 있는 메타데이터의 종류와 형식에 구애받지 않는다: 많은 수의 메타데이터 속성과 연관된 항목이 있을 수 있으며, 그렇지 않은 항목도 있는데, 같은 속성이 모든 항목에 적용되는 것이 아니라, 특정 항목에만 속성을 연관시킬 수 있다.

4. 구현

본 연구에서는 지금까지 설명한 계층적 파일 시스템을 DBMS로 표현하고, 메타데이터를 관리하는 기법을 이용하여 실제 리눅스 파일 시스템 상에 구현하여 보았다.

구현 환경은 다음과 같다:
- 운영체제: White Box Linux 4 (Linux 2.6.9)
- RDBMS: MySQL 4.0.20
- 구현 언어: Python 2.4.3
- 데이터베이스 접근 모듈: MySQLdb[6]

다음은 MySQL에서 테이블 생성을 위한 SQL 선언이다:

```
CREATE TABLE  fs (  # 파일 시스템 계층구조
  pinode  INT,
  inode   INT,
  mtime   INT,
  ctime   INT,
  ftype   CHAR(2),
  fsize   INT,
  name    CHAR(255),
  INDEX   inode_idx (inode),
  INDEX   pinode_idx (pinode),
  INDEX   mtime_idx (mtime)
);

CREATE TABLE md ( # 메타데이터
  inode   INT,
  attr    CHAR(100),
  value   CHAR(255),
  INDEX   inode_idx (inode),
  INDEX   attr_idx (attr)
);
```

테스트에서 파일 시스템의 루트 디렉터리(/) 이하 338,924 항목(약 90GB)을 데이터베이스에 입력하였다. 성능 비교를 위해 유닉스 계열에서 사용되며 파일시스템에서 파일 위치를 인덱싱하여 검색할 수 있도록 하는 slocate 유틸리티([5])와 비교하였다.

표 1. slocate와 성능 비교 결과

반복		slocate	본 연구 구현
1	검색된 항목	111개	91개
	소요 시간	0.161s	1.041s
	항목당 소요시간	**1.45us**	**11.44us**
2	검색된 항목	4585개	6543개
	소요 시간	0.390s	4.240s
	항목당 소요시간	**0.085us**	**0.648us**
3	검색된 항목	50682개	26678개
	소요 시간	3.456s	3.267s
	항목당 소요시간	**0.068us**	**0.122us**

본 연구에서 구현물은 인터프리터 방식의 파이썬 언어로 구현하였으므로 성능 면에서는 slocate 보다 떨어지는 것

을 확인 할 수 있었다. 그러나, 데이터베이스를 이용하기 때문에 항목의 수가 많아질수록 성능 면에서 차이가 줄어드는 것을 확인 할 수 있다.

그림 8은 본 연구에서 구현된 프로그램으로 특정 파일을 파일 이름을 이용하여 검색한 결과이다. 파일 이름으로 SQL 질의를 하고 결과로 파일의 경로를 출력하도록 구현하였다.

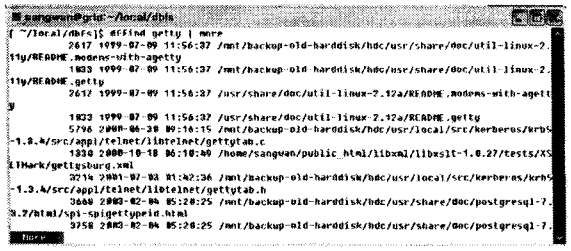


그림 8. 파일 검색 출력 결과

5. 결론 및 향후 연구

본 연구에서는 데이터베이스를 이용하여 계층적 파일시스템에 적용할 수 있는 메타데이터 관리 방법을 설계하고 일부를 구현하였다. 계층적 파일 시스템은 오늘날 거의 모든 범용 컴퓨터 시스템에 사용되어 있는데, 디렉터리와 파일을 체계적으로 구성할 수 있고, 직관적이고도, 단순하다는 장점이 존재한다. 그러나 인덱싱이 되지 않을 경우 검색이 용이 하지 않고, 분류 기준에 계층적 관계가 없을 경우 분류가 어렵고, 메타데이터를 관리할 수 없다는 단점이 존재한다.

본 연구에서 설계한 메타데이터관리 방법은 계층적 파일 시스템의 계층 구조를 RDBMS로 표현을 하고, 메타데이터를 관리하기 위해 속성, 값의 쌍을 테이블로 관리함으로써 RDBMS의 장점을 이용하여 검색을 쉽게 하고, 메타데이터의 확장을 가능하게 한다는 특징이 있다.

그러나, 운영체제의 변경 없이 파일 시스템이 정보만을 이용하여 DB를 구성하기 때문에 파일 시스템의 변경된 내용이 곧바로 DB에 반영되지 않는 다는 단점도 존재한다. 파일 시스템의 변경 사항을 실시간으로 DB로 반영하기 우해서는 운영체제나 파일 시스템이 변경되어야 한다.

파일 시스템의 메타데이터를 데이터베이스에 저장하는 것은 많은 장점이 있다. 지리적으로 분산된 대용량의 저장장치를 연동하여 사용하는 데이터 그리드 분야에서는 분산된 데이터의 복제와 검색이 빈번하게 일어나는데, 이 때 마다 파일 시스템을 일일이 검사하는 것은 비효율적이다. 따라서 데이터베이스의 빠른 검색 기능을 이용하여 복제된 데이터의 위치와 메타데이터를 관리하면 상당히 효율적인 데이터 관리가 가능하게 된다.

참고 자료

[1] Wikipedia.org: Hierarchical File System
http://en.wikipedia.org/wiki/Hierarchical_File_System
[2] DBFS
http://tech.inhelsinki.nl/dbfs/
[3] Managing Hierarchical Data in MySQL
http://dev.mysql.com/tech-resources/articles/hierarchical-data.html
[4] Understanding Metadata
http://www.niso.org/standards/resources/UnderstandingMetadata.pdf
[5] Secure Locate
http://slocate.trakker.ca/
[6] MySQL for Python
http://sourceforge.net/projects/mysql-python