FMAX(Flexible MetadatA eXtention) File System: 효과적인 메타데이터 관리를 위한 파일 시스템 모델

박치현° 노홍찬 유진희 임지영 김우철 박상현
연세대학교 컴퓨터과학과
{tianell, fallsmal, jhyou, jylim, twelvepp, sanghyun}@yonsei.ac.kr

FMAX File System : A File system model for effective metadata management

Chihyun Park° Hongchan Roh Jinhee You Jiyoung Lim Woocheol Kim Sanghyun Park
Dept. of Computer Science, Yonsei University

요 약

방대한 양의 데이터를 처리하는 요구가 증가하면서 그로부터 유용한 정보를 추출해낼 수 있는 방법에 대한 필요성이 증가하고 있다. 메타데이터(metadata)를 관리하는 파일 시스템은 이러한 일을 빠르고 효과적으로 수행할 수 있는 방법 중 하나이다. 본 논문에서는 통합적인 메타데이터 관리를 가능하게 하고, 임베디드(embedded) 환경으로도 확장이 가능한 새로운 파일 시스템을 제안 한다. 이를 위해 기존 리눅스(Linux) 파일 시스템을 바탕으로 새로운 파일 시스템을 설계하고 제시한다. 이렇게 설계된 파일 시스템은 사용자와 응용프로그램에게 데이터의 다양한 뷰(view)를 제공할 수 있고 시스템 상에 존재하는 많은 양의 데이터를 통합하여 효율적으로 관리 할 수 있다.

1. 서론

최근 컴퓨터 시스템에서 저장 공간 크기의 증가로 다양한 종류의 많은 파일들을 저장할 수 있게 되었다. 이에 따라 방대한 양의 파일들을 효율적으로 관리해야 하는 필요성이 증가하고 있고, 메타데이터[1]의 사용은 이러한 파일들을 효율적으로 관리할 수 있는 방법 중의 하나이다.

최근 메타데이터의 관리와 활용에 대한 연구는 다음과 같이 다양한 분야에서 이루어지고 있다.

[1] 에서는 메타데이터를 이용한 시멘틱 파일 시스템(semantic file system)에 대한 연구가 진행되었다. 이 논문은 인터넷 응용프로그램에서 현재 많이 사용되고 있는 태깅(tagging)을 이용하여 그래프 형태로 메타데이터 정보를 제공하는 TagFS(Tag File System)를 제안하였다. 태깅이란 파일들을 분류화할 수 있도록 파일에 어떤 정보를 추가하는 것으로 태그(tag)로 표현되는 메타데이터를 추가하는 것이다. [1] 에서는 이러한 태그 정보를 RDF(Resource Description Framework)에 저장하고 제공함으로써 많은 종류의 파일들을 효과적으로 관리하도록 하였다.

[2]에서는 메타데이터를 분류화하는 연구가 진행되었고, [3]에서는 데이터베이스 관리 시스템에 기반하여 메타데이터를 관리하는 방법들에 대한 연구가 진행되었다. 또 [4]에서는 웹 기반의 메타데이터 파일 시스템에 대한 연구가, [5]에서는 전통적인 파일 시스템상에서 메타데이터를 관리하는데 생기는 문제점에 관한 연구가 진행되었다. [6], [7]에서는 대규모 분산 메타 파일 시스템에 관한 연구가 이루어졌다.

이와 같이 기존의 연구들은 다양한 분야에 걸쳐 메타데이터를 적용하였지만 단일 파일 시스템 내에서 통합적으로 메타데이터를 관리하는 연구는 [1] 을 제외하고는 활발히 이루어지지 않고 있는 상황이다. 따라서 본 논문에서는 단일 파일 시스템 내에서 대규모 파일을 효율적으로 관리하기 위하여 메타데이터를 통합적으로 관리할 수 있는 새로운 파일 시스템을 제안한다.

본 논문은 [1]에서 제안한 파일 시스템과는 다르게 메타데이터를 추출할 수 있는 MGM(Metadata Generation Module)이라는 부분을 제안한다. 또한 메타데이터의 효율적인 관리를 위해 논리적인 부분과 물리적인 부분으로 나누어서 메타데이터 관리를 제안한다. 또한 메타데이터를 제공하는 방법도 [1]에서 제안한 태그를 포함한 그래프 형태가 아니라 메타데이터 파일 형태로 제공하고 그것을 바탕으로 멀티 디렉토리(multi directory)같은 형태로 활용할 수 있도록 한다. 마지막으로 리눅스 커널(kernel) 코드 수정을 구현 방법으로 하여 모바일(mobile) 기기 등 다른 시스템으로의 확장 가능성이 높은 파일 시스템을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 리눅스 파일 시스템의 전체적인 구조를 본 논문과 관련 지어 설명한다. 3장에서는 구체적으로 본 논문이 제안하는 파일 시스템을 기술하는데 MGM을 이용한

[1] 메타데이터: 데이터에 관한 데이터를 의미하는 것으로서 데이터를 사용, 관리, 이해하는 데 편의성을 제공하는 것이다.

메타데이터를 추출 과정, 논리적·물리적 파일을 통한 메타데이터 관리 방법, 끝으로 메타데이터의 활용 방법 중 하나인 멀티 디렉토리를 기술한다. 마지막으로 4장에서는 결론과 향후 발전 방안을 서술 한다.

2. 전체 리눅스 파일 시스템 구조

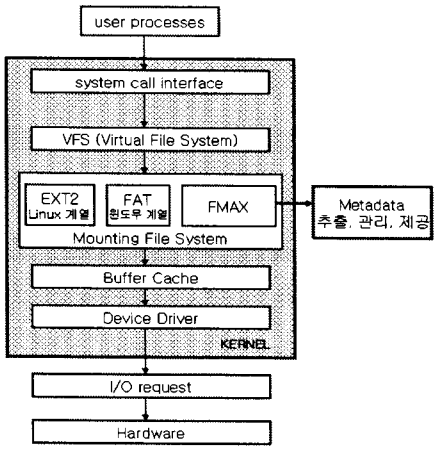

그림 1. 리눅스 파일 시스템

그림 1은 일반적인 리눅스 파일 시스템의 구조를 나타낸 그림이다. 그림 1을 보면 여러 부분으로 파일 시스템이 구성되어있는데 본 논문에서 중점을 두어 다루는 부분은 가상파일시스템(Virtual File System)영역과 여러 파일 시스템을 적재(mount)할 수 있는(Mounting File System)영역이다. 전체 파일 시스템에서 이 두 부분만 중점적으로 다루는 이유는 본 논문에서 제안하는 파일 시스템은 FAT이나 EXT2같은 하나의 파일 시스템의 형태로 모듈화되기 때문이다. 따라서 다른 영역은 보통의 파일 시스템이 사용하는 것과 동일하게 접근하고 사용한다. 이렇게 설계를 하는 것의 장점은 기존 리눅스 파일 시스템을 사용할 수 있는 어떠한 시스템에도 적용이 가능하며 메타데이터를 관리하는 파일 시스템을 사용하면서 동시에 전통적인 파일 시스템 또한 사용 가능하기 때문에 파일 관리에 있어 좀 더 유연해 질 수 있다는 것이다. 이러한 이유로 본 논문에서 제안하는 파일 시스템을 FMAX(Flexible MetadatA eXtention) 파일 시스템 이라고 한다.

3. FMAX 파일 시스템

3.1 FMAX의 구조와 전체적인 작업 흐름

FMAX는 기존 파일 시스템이 수행해 왔던 파일에 대한 연산들을 모두 제공해주면서 추가적으로 메타데이터를 활용하는 새로운 기능을 제공해준다. 또한 FMAX의 기본적인 설계는 리눅스 기반의 FAT계열 파일 시스템을 바탕으로 한다. 이렇게 하는 이유는 기존 시스템과의 호환성을 유지할 수 있는 방법이기 때문이다. 따라서 FMAX의 구성은 크게 기존 파일 시스템의 연산들을 수행해줄 수 있는 영역과 메타데이터에 관한 작업을 해줄 수 있는 영역으로 나누어져 있다. 메타데이터에 관한 주요 작업은 크게 3가지로 나눌 수 있는데 다음과 같다.

- 메타데이터의 추출: MGM 이라는 모듈이 메타데이터를 추출한다. 이 모듈이 하는 일은 기존 응용프로그램이 담당하던 메타데이터 추출 기능을 모듈로 분리한 후 파일 시스템 차원에서 이 일을 담당한다.
- 메타데이터 관리: 메타데이터의 효율적인 저장을 위한 논리적인 스키마를 설계하였고 또한 저장의 효율성을 높이기 위해 논리적인 스키마로 구성된 파일을 물리적으로 저장할 때는 분할된 파일 형태로 저장을 한다.
- 메타데이터 활용: 멀티 디렉토리처럼 메타데이터를 사용하여 사용자와 응용프로그램에게 데이터 구조의 다양한 뷰(view)를 제공할 수 있다.

위에 언급한 내용은 다음 소 단원에서 각각 자세히 기술하겠다. 위의 내용을 정리해서 FMAX의 구조를 그림으로 나타내면 그림 2와 같다. 크게 일반 파일시스템 영역과 FMAX파일 시스템 영역의 두 부분으로 나눌 수 있으며 메타데이터의 활용을 위해 새로운 함수들이 추가되었고 파일 종류별로 그런 기능들이 수행되도록 나누었다. 또한 멀티 디렉토리 부분을 하단에 두어서 일반 파일 시스템 영역과 FMAX 파일 시스템 영역 모두 메타데이터의 활용을 제공받을 수 있도록 했으며 메타데이터의 관리를 위해 I/O 함수를 정의 하였다.

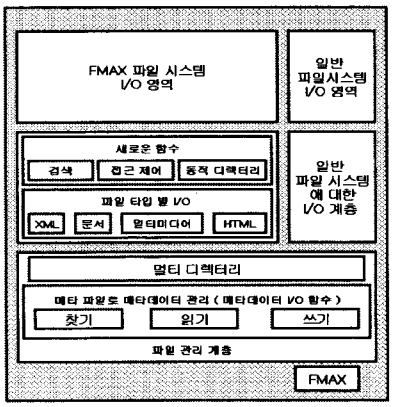

그림 2. FMAX 내부 구조

그림 2와 같은 내부구조를 갖는 FMAX는 그림 3과 같은 전체적인 작업 흐름을 거쳐서 파일 시스템으로 동작하게 된다. 실제 파일 시스템상에서 발생하고

요청되는 작업은 많지만 그림 3에서는 설명의 편의를 위해 파일 시스템에서 수행되는 작업을 단순화 시켜서 전체적인 흐름을 나타냈다.

예를 들어 휴가철 찍은 사진들을 컴퓨터에 저장한다고 하자. 그럼 그림 3의 작업흐름에 따라서 요청 처리 프로세서는 '새로운 파일 저장 요청'을 수행하고 메모리 카드에 있는 사진들을 내 컴퓨터의 저장소에 저장하게 되는데 파일들을 저장하기 전에 MGM체인을 통과한다. 그럼 각 파일들에 대한 메타데이터들이 생성되고 구조화된 메타데이터 파일 저장소에 저장이 된다. 그리고 예를 들어 저장된 사진들에 대해서 날짜 별로 찍은 사진들을 보고 싶다는 '파일 접근 요청'하게 되면 하나의 디렉터리 안에 특별한 분류 없이 저장되었던 사진 파일들에 대해서 날짜 별로 멀티 디렉토리를 생성해준다. 이때 우리가 생성하고 저장해 놓은 메타데이터를 사용해서 멀티 디렉토리를 구성한다. 마지막으로 사용자에게 날짜 메타데이터를 사용해서 만들어진 멀티 디렉토리 뷰를 제공한다.

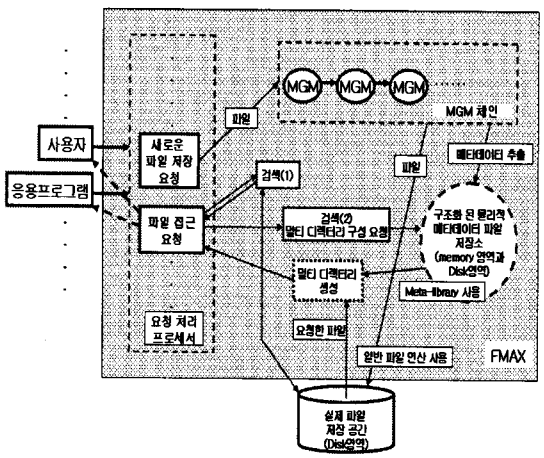

그림 3. FMAX 파일 시스템에서의 전체 작업 흐름도

3.2 MGM(Metadata Generator Module)의 설계

메타데이터를 관리하는 파일 시스템을 설계할 때 가장 먼저 연구했던 부분이 파일들에서 어떠한 메타데이터를 추출할 것이고, 메타데이터를 추출하는 구조를 어떻게 설계할 것인가에 대한 것이었다. 본 논문에서는 체인 형태로 MGM을 만들어서 메타데이터를 추출하는 방법으로 설계를 하였다. MGM은 그림 4에서 보듯이 한 개가 아니다. 여러 MGM들이 체인으로 연결되어있는데 각각의 MGM하나는 우리가 추출하고자 하는 메타데이터의 한 종류를 추출할 수 있는 단위이다. 즉 MGM의 예로는 사진파일의 크기, 문서파일의 생성 시간 등이다. 이 각각이 하나의 MGM의 단위가 되고 MGM들은 체인 구조 즉 리스트 구조이기 때문에 동적으로 추가와 제거가 가능하다. MGM 체인에 메타데이터를 추출해야 하는 파일이 입력으로 들어오면 현재 구성되어 있는 MGM 체인 안의 모든 모듈들을 지나가면서 추출할 수 있는 메타데이터를 추출하도록 한다. 이렇게 체인을 지나갈 동안 추출된 메타데이터들은 메타데이터 저장소에 저장된다.

이런 구조를 갖는다면 MGM의 수가 많아 질수록 메타데이터의 추출에 과부하가 생길 수 있으며 입력 파일이 메타데이터 추출에 필요하지 않은 MGM도 지나가야 하는 문제점들이 발생할 수 있다. 하지만 이미 생성된 MGM은 다른 메타데이터를 추출하는데 재활용될 수 있고 사용자가 파일 종류에 따라 동적으로 메타데이터를 생성할 수 있다는 이점이 있다. 그림 4는 MGM의 구조에 대한 그림이다.

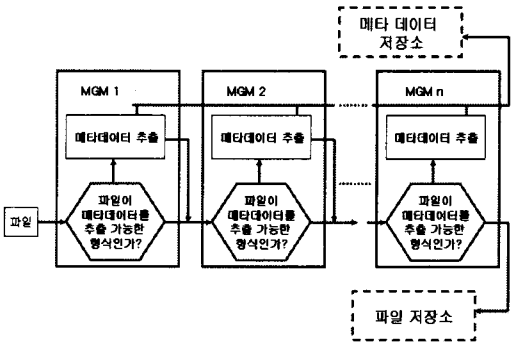

그림 4. MGM체인 구조도

3.3 생성된 메타데이터 저장과 관리

MGM 체인을 통과하면서 추출된 메타데이터들은 메타데이터 저장소에 저장이 된다. 여기서 저장소란 메타데이터를 저장할 수 있게 설계된 파일을 말한다. 메타데이터의 저장과 관리 과정은 실제로 물리적 저장 공간에 파일 형태로 메타데이터를 저장하는 과정을 말하지만, 본 논문에서는 저장 과정을 저장할 메타데이터 내용을 논리적 스키마로 구조화하는 과정과 그에 대응되는 물리적인 파일로 저장하는 두 부분으로 나누어서 제안한다. .

메타데이터를 논리적 스키마로 구조화하는 이유는 MGM이 추출한 메타데이터를 체계적으로 관리 할 수 있는 하나의 방법이기 때문이다. 표 1은 논리적인 구조를 테이블 형태로 표현한 것이다.

MGM 별로 추출된 메타데이터는 표 1에서 보면 알 수 있듯이 예를 들어 MGM(FILEINFO), MUSIC(MOVIE) 라는 값과 Tag라는 속성으로 표현이 된다. 여기에 부가적으로 다른 속성들을 추가해서 논리적 스키마 구조로 만든다. 여기서 MGM(FILEINFO)라는 값이 의미하는 바는 어떤 파일이 들어왔을 때 파일 정보에 관한 메타데이터를 추출할 수 있다는 것이고 파일 정보에 관한 메타데이터들은 예를 들어 파일 크기가 될 수도 있고 생성 시간이 될 수도 있다. 이러한 내용은 Tag 속성으로 표현이 되고 실제 그 값은 value(s)속성에 저장 된다.

표 1. 메타데이터 저장 구조의 논리적인 스키마

time	Primary Path	file	MGM	Tag	value(s)
20070501 4:03	/music/ jazz	j.mp3	MGM (FILEINFO)	FILE_ SIZE	650 MB
20070501 4:03	/music/ jazz	j.mp3	MGM (MUSIC)	PLAY _TIME	3' 25"
20070503 10:30	/movie/ action	a.avi	MGM (FILEINFO)	FILE_ SIZE	723 MB
20070503 10:30	/movie/ action	a.avi	MGM (MOVIE)	PLAY _TIME	57' 24"
20070503 10:30	/movie/ action	a.avi	MGM (MOVIE)	GENR E	Action , SFX

여기서 태그화를 하는 과정이 [1]에서 나온 태깅 과정과 비슷하다고 생각할 수 있는데 근본적으로 두 방법은 차이가 있다. [1]에서의 태깅은 하나의 메타데이터 자체를 태그로 생각해서 해당 파일에 연결시키고 그래프 형태로 제공하는 것이고, 본 논문에서의 태그는 단지 얻어지는 메타데이터들의 타입을 말하는 것이다. 본 논문에서의 실제 메타데이터는 value(s) 속성에 들어있다.

그런데 위와 같은 논리적인 스키마를 갖는 테이블 형태의 파일을 그대로 하나의 파일로 저장 한다면 메타데이터의 관리가 비효율적이 될 수 있다. 이유는 메타데이터를 추출해야 하는 파일의 종류가 많아 질수록 메타데이터의 양도 증가하고 따라서 인덱스를 사용한다 하더라도 효율적인 탐색과 관리가 어려워지기 때문이다. 그러나 위와 같은 논리적 스키마는 유사한 메타데이터들을 통합 관리하기 때문에 중복 저장을 피할 수 있고 서로 다른 응용프로그램에서 공유할 수 있다는 장점이 있다. 기존 파일 시스템에서는 각각의 응용프로그램 단계에서 메타데이터를 추출하고 관리하였기 때문에 메타데이터들을 공유하기 어려웠고 중복 저장되는 메타데이터도 많았다. 따라서 표 1과 같은 논리적인 구조를 유지하면서 메타데이터 관리의 효율성도 높일 수 있는 방법을 연구하였고 실제 물리적으로 메타데이터 파일이 저장될 때는 표 1의 논리적인 구조를 유지한 채 저장과 차후 검색에 효율적인, 분산 파일의 형태로 저장되도록 설계하였다.

효율적인 메타데이터 저장 파일의 분할을 위해 본 논문에서 제안하는 방법은 논리적 스키마를 갖는 하나의 파일을 분할할 때 Global_MGM_File 과 Global_Tag_File, Local_Metadata_File로 나누어서 저장하는 것이다. 이렇게 3가지 파일로 분할한 기준은 각 분할된 파일 별로 최대한 중복되는 내용을 줄여서 메타데이터의 검색의 효율성을 높이도록 하는 것이다. Global_MGM_File은 표 2와 같은 테이블 구조이며

표 2. Global_MGM_File의 구조

TagID	MGM	tag
1	MGM(FILEINFO)	FILE_SIZE
2	MGM(MUSIC)	PLAY_TIME
3	MGM(VIDEO)	PLAY_TIME
4	…	…

표 3. Global_Tag_File의 구조

TagID	File_Names
1	/music/jazz/j.mp3 /movie/action/a.avi
2	/music/jazz/j.mp3 /music/hiphop/b.mp3
3	/movie/action/a.avi
4	….

표 4. j.mp3 파일에 해당하는 Local_Metadata_File 구조

TagID	time	values
2	2007-05-01 2007-06-21	3' 25" 2' 34"
5	2007-05-02	5.1 MB
…	…	…
File path: /music/jazz/.j.mp3.meta		

3가지 속성이 있다. Global_MGM_File이 갖는 의미는 논리적 스키마를 갖는 메타데이터 저장 파일에서 중복적으로 나오는 MGM들과 그에 대응하는 태그들을 실제 저장되는 파일 구조에는 한번만 나타나도록 해준다는 것이다. TagID라는 속성을 키(key)값으로 하여 나타낼 수 있다.

표 3은 Global_Tag_File의 구조를 나타낸다. 특징은 File_Names라는 속성이 있는데 하나의 TagID에 속하는 File_Names는 여러 개가 될 수 있다. 이유는 서로 다른 파일이라고 할지 라도 같은 MGM을 통해서 같은 종류의 메타데이터를 추출할 수 있기 때문이다.

마지막으로 Local_Metadata_File에 대하여 살펴보겠다. 앞에서 언급한 두 개의 메타데이터 파일들과 Local_Metadata_File의 차이점이 있다. 앞에 언급한 두 파일은 전체 FMAX에서 하나씩만 존재하는 것이다. 그러나 Local_Metadata_File은 파일 시스템에서 저장되는 각각의 파일들에 대해서 하나씩 만들어 진다. Local_Metadata_File의 내부 구조는 테이블형식으로 되어있으며 해당 파일에 대한 메타데이터의 저장이 요청되는 순서대로 Time과 Values 속성 들이 한 개 이상 삽입된다. 즉 실제 파일에 대한 메타데이터 값들이 저장된다. 표 4는 Local_Metadata_File의 구조를 나타낸다.

3.4 메타데이터의 활용

멀티 디렉토리는 본 논문이 제안하고 있는 대표적인 메타데이터의 활용 방안이다. 멀티 디렉토리 개념은 [6]에서 제안한 파일과 디렉토리에 대한 자동화된

인덱싱에서 아이디어를 얻었다. [6]에서는 자동화된 인덱싱 방법을 사용하면 전통적인 트리 구조의 파일 시스템보다 파일 관리와 저장 효율성이 높다고 한다. 메타데이터를 활용하면 접근하려는 파일과 디렉토리에 대한 자동화된 인덱스를 더욱 빠르고 효율적으로 구성할 수 있다.

멀티 디렉토리는 동적 디렉토리 뷰(Dynamic Directory View)라고 부를 수 있으며 사용자 계층이나 응용프로그램 계층에서 파일 접근 요청을 할 때 메타데이터를 활용해야 하는 요청이라고 분석이 되고, 그 중 멀티 디렉토리를 제공해야 한다고 분석되면 만들어지게 된다. 이 과정은 3.1에서 예시로 들었던 내용과 그림 3을 보면 알 수 있다.

이제 멀티 디렉토리를 구성하는 방법에 대하여 알아보자. 파일에 대한 접근 요청이 들어올 때 그 요청에서 필요하다고 판단되는 경로에 대해서만 사용자나 응용프로그램에게 제공해주는 것으로 현재 구성되어있는 트리 구조의 파일과 디렉토리 구조를 바탕으로 새로운 파일 구조를 만들어서 제공해야 한다. 리눅스 파일 시스템에서 파일과 디렉토리들은 dentry 객체와 연관이 되어 있다.

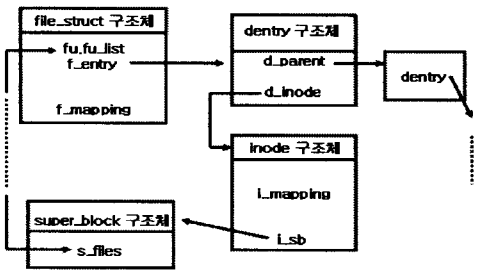

그림 5. 리눅스 파일 시스템에서의 주요 구조체 관계

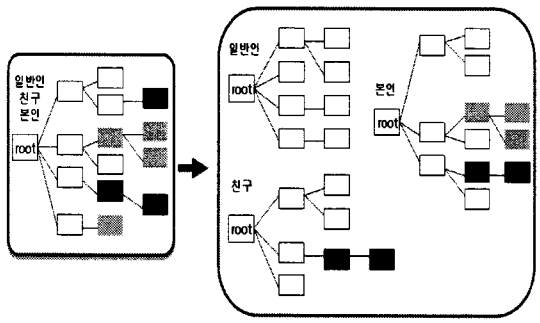

그림 6. 멀티 디렉토리의 모습

그림 5와 같은 구조로 파일 시스템에서 파일과 디렉토리는 연결 되어있다. 하나의 파일은 그것의 정보를 나타내는 file_struct 구조체가 있으며 그것은 현재 그 파일이 저장되어 있는 디렉토리를 나타내는 dentry 구조체를 포인터로 가리키고 있다. dentry 구조체는 inode 구조체를 또 포인터로 가리키며 inode 구조체에는 디스크상에서 실제 물리적인 파일의 저장에 관련된 정보들이 들어있다. 위와 같은 구조로 파일이 저장되어 있기 때문에 멀티 디렉토리를 제공하려면 현재 이런 연결로 되어있는 부분을 우리가 원하는 연결로 바꾸어야 한다. 멀티디렉토리를 위해서는 그 중 dentry 구조체의 수정이 필요하다. 즉 우리가 멀티 디렉토리를 구성하는데 필요한 메타데이터를 가지고 논리적으로 어떻게 디렉토리 구조를 변경하여 제공할 것인지 정의를 한 후 그것에 맞게 dentry 구조체의 연결(포인터)을 변경한다. 그림 6은 멀티 디렉토리 방법으로 파일을 재구성하여 사용자나 응용프로그램에 제공하는 것을 나타낸다.

4. 결론

본 논문에서는 리눅스 기반의 메타데이터를 관리할 수 있는 새로운 파일 시스템에 대하여 제안하였다. 또한 확장성이 높은 리눅스 기반으로 모듈화된 파일 시스템을 제안 함으로서 향후 플래쉬 메모리(flash memory)나 리눅스 파일 시스템이 동작하는 다양한 시스템에 적용 가능하도록 하는 것을 목표로 하였다.

이러한 파일 시스템을 위해 MGM이라는 메타데이터를 추출 부분과, 메타데이터 저장과 관리를 위한 논리적 스키마 형태와 이것의 저장과 관리의 효율성을 위하여 분할된 형태의 저장구조를 제시하였다. 또 메타데이터의 활용 방안 중 대표적인 예로 멀티 디렉토리에 대하여 설명하고 제시하였다.

향후 메타데이터의 저장과 관리에 관련한 구조들의 효율성을 최대화 하기 위한 연구를 수행할 예정이며 메타데이터의 활용 범위를 넓히기 위해 멀티 디렉토리에 기반한 보안 기능에 관한 연구와 사용자와 응용프로그램 계층에서 메타데이터를 활용할 수 있는 방법들을 연구할 예정이다.

참고 문헌

[1] Simon Schenk, Olaf G¨orlitz, Steffen Staab, "TagFS: Bringing Semantic Metadata to the Filesystem", Demos and Posters of the 3rd European Semantic Web Conference(ESWC 2006), Budva, Montenegro, 11th 14th June 2006

[2] Ganesan Shankaranarayanan, Adir Even, "The metadata enigma", Communications of the ACM, Volume 49 Issue 2, pp.88~94 February 2006.

[3] Divesh Srivastava, Yannis Velegrakis, "Storage engine and access methods: Intensional associations between data and metadata", Proceedings of the 2007 ACM SIGMOD international conference on Management of data SIGMOD '07, pp.131~136, June 2007.

[4] Bernhard Schandl, Ross King, "The SemDAV Project: Metadata Management for Unstructured Content", Proceedings of the 1st international workshop on Contextualized attention metadata: collecting, managing and exploiting of rich usage

information CAMA '06, pp.27~31, November 2006.
[5] Gregory R. Ganger, Marshall Kirk McKusick, Craig A. N. Soules, Yale N. Patt, "Soft updates: a solution to the metadata update problem in file systems", ACM Transactions on Computer Systems (TOCS), Volume 18, Issue 2, pp.127~153, May 2000.
[6] David K. Gifford, Pierre Jouvelot, Mark A. Sheldon, James W. O'Toole, "Semantic file systems", ACM SIGOPS Operating Systems Review , Proceedings of the thirteenth ACM symposium on Operating systems principles SOSP '91, Volume 25, Issue 5, pp.16~25, September 1991.
[7] 차명훈, 이상민, 김준, 김영균, 김명준, "대규모 분산 파일 시스템 환경의 메타 데이터 관리", 전자통신동향분석 제22권, 제3호, 154쪽~165쪽, 2006년 6월.